A Method of On Road Vehicle Tracking

Kaiyuan Yao

A thesis submitted for the degree of Master of Philosophy

The University of Edinburgh

May 2008

Abstract

Vehicle detection and tracking systems, which could be used for driver assistance continue be to an area of active research. This thesis presents a method of tracking which could be used as part of a driver assistance system for collision avoidance. It uses images taken by a pair of stereo cameras mounted on a vehicle.

A review of existing approaches to vehicle detection and tracking and previous approaches to stereo imaging are presented. The benefits to be gained from deriving u-disparity and V-disparity maps from stereo image pairs are described. Vehicles are segmented using information from both u-disparity and V-disparity maps. This U-V-disparity based approach is then combined with a condensation algorithm for on-road vehicle tracking. The segmentation results are then used to predict vehicle motion instead of using a pre-learned dynamic motion model. The condensation algorithm is modified to suit this method of motion prediction. As for the conventional condensation algorithm, particles generated from the motion prediction are weighted by the similarity of colour histograms before the sampling process.

Results from testing this method on a stereo sequence taken in a complex urban road scene with different vehicle types and motions are presented. Successful tracking of multiple vehicles with multiple motions is demonstrated.

Acknowledgements

Firstly, I would like to thank both my supervisors, Dr John Hannah and Dr David Renshaw. During the 2 years of my study, they gave me great support. Dr Hannah gave advice and guidance not only on my research but also my living in the UK.

Thank Dr Peter Hillman for various discussions on camera calibration and stereo camera topics. I would also like to thank Paul Kuo who helped me a lot in understanding basic image processing theories at the beginning of the study.

Thanks also go to Dr Michele Zanin from TeV Technologies of Vision Team, who generously provided DIPLODOC data for me as my test data of my project.

Last but not least, I would like to thank my parents who worked very hard to support my study in the UK.

Content

List of Figures

List of Equations

List of Tables

List of Abbreviations

BSCE	**Basic Stereo Consistency Event**
DOF	**Degree of Freedom**
GRF	**Gibbs Random Field**
ICM	**Iterated Conditional Modes**
MAP	**Maximum A Priori**
MFT	**Mean Field Theory**
MMD	**Modified Metropolis Dynamics**
MRF	**Markov Random Field**
RANSAC	**Random Sample Consensus**
SAD	**Sum of Absolute Difference**
Voxel	**Volume Pixel**

List of Symbols

M	object
m	projection in image
(u,v)	image pixel coordinates
d	disparity
(X,Y,Z)	camera coordinates
F	a set of randoms
g	random value
$P()$	probability distribution
$U()$	clique energy function
$V()$	clique potential
cq	clique
Ne	neighbourhood
π_i	particle weights
s	sample sets
f	focal length
c	principle point
α	pixel skew coefficient
k	camera distorsion
K	camera calibration matrix
t	stereo baseline length
π	constant
μ	mean
σ	variance
En	energy
(x,y)	bounding box position
w	bounding box width
h	bounding box height
A	motion model tuning parameter
$h()$	colour histogram function
p_i	particles
cu	cumulative probability
N	number of particles
R	Gaussian random number

E	**estimation of probability distribution**
β	**bounding box horizontal movement between frames**

1 Introduction

In 2005, according to the Department for Transport, the annual report on road casualties in Great Britain claims 3,201 people were killed on Britain's roads. The number of people seriously injured was 28,954, and total casualties in 2005 were 271,017. Road safety is related to everyone's daily life. Efforts have therefore been made to improve the road safety. Automotive industries and researchers have tried to develop driver assistant systems to avoid accidents due to human error, which is the main source of accidents.

Driver assistance systems usually can be divided into two types: driving environment perception and driver activity perception. The driving environment perception systems usually use passive (optical cameras) or active (radar and lidar) devices to gain information about the environment surrounding the vehicle and the information is processed to provide a better understanding of the environment for the driver. While the driver activity perception system usually uses optical cameras to monitor the driver's activity and decide if the driver is in a good condition to drive. The driving environment perception includes a wide range of applications, such as lane detection and following, vehicle detection and tracking, etc.

This project aims to develop a vehicle detection and tracking system which can be used as a part of a driver assistance system. The desired output of the system is to detect and track vehicles in the image sequences taken by a pair of stereo cameras mounted on a vehicle. By tracking in the stereo sequence, the results could be used further for a driver assistance system such as collision avoidance.

There are two main challenges for this project. First, since the sequence is taken by a pair of moving cameras, some popular tracking methods may not work well. Second, the sequence is taken in an urban area which involves multiples types of vehicle and complex situations. For the first challenge, tracking with moving cameras is very

different from tracking with static cameras. The background is also changing due to the camera moving. This makes the object detection and segmentation more difficult. Also, the object motion is not its own motion but that with respect to the cameras (driver's vehicle). This makes the prediction of object motion more difficult. The second challenge requires the tracker to be able to track multiple objects and handle possible situations such as objects emerging and disappearing and also occlusion.

The approach adopted uses a combined colour and U-V-disparity condensation tracker to fulfil this tracking task and conquer the challenges. U-V-disparity is a special way of using the information from a sparse disparity map. It concentrates sparse disparity points onto horizontal and vertical directions and makes one-dimensional segmentation possible. Vehicles can be segmented using a rectangular bounding box. This U-V-disparity approach is used for vehicle detection and thus a prediction of vehicle positions. The U-V-disparity detection is used to predict the position of the vehicle instead of introducing a sophisticated motion model for different vehicles and different motion types. This decreases the number of parameters which need to be tuned for the motion prediction and makes the tracker more general for different types of vehicles and motions. Colour distributions are modelled as colour histograms and are used as observation models. At each time step, the current colour histogram for the detected vehicle is compared with its previous colour histogram by the Bhattacharyya distance, which is used as sample weights for a condensation algorithm. The condensation algorithm adopted is slightly different from normal ones. The algorithm is modified to suit the U-V-disparity motion prediction. The algorithm differs from the normal condensation algorithm, as it first detects the vehicles and generates particles for them, calculates weights, then samples according to the weights and finally takes the weighted mean as output. The particles are generated partially from the U-V-disparity prediction and partially from motion prediction from the previous position. Weighting of the particles is done before the sampling, which makes the final results more close to the real position.

This modification decreases the problem of degeneration in the normal condensation algorithm, which uses the same sets of particles and purifies the particles just by motion modelling and sampling.

A brief overview of the theoretical background including stereo, U-V-disparity and condensation algorithms is given in **Chapter 2**. **Chapter 3** describes some preparatory work of the project which leads to final approach. **Chapter 4** gives a detailed description of the algorithms, including the normal condensation algorithm and the modified one. Results and comparison of the normal and modified algorithm can be found in **Chapter 5**. Finally, conclusions and future work are presented in **Chapter 6**.

2 Theoretical Background

2.1 Introduction

In this chapter, some basic theoretical background is reviewed together with related literature review. First, the basics of stereo are illustrated with some figures. Following the stereo theory, papers on revealing the 3D information based on stereo techniques are reviewed. Following the stereo techniques, the related papers of camera calibration and multi-camera approaches are reviewed. As a unique way to process the sparse disparity map, U-V-disparity methods and papers are introduced. It is also compared with Markov Random Fields theory from an image/object segmentation perspective. Finally, Condensation/Particle Filtering methods are discussed as they are the most popular tracking methods in the existing literature.

2.2 Stereo Vision

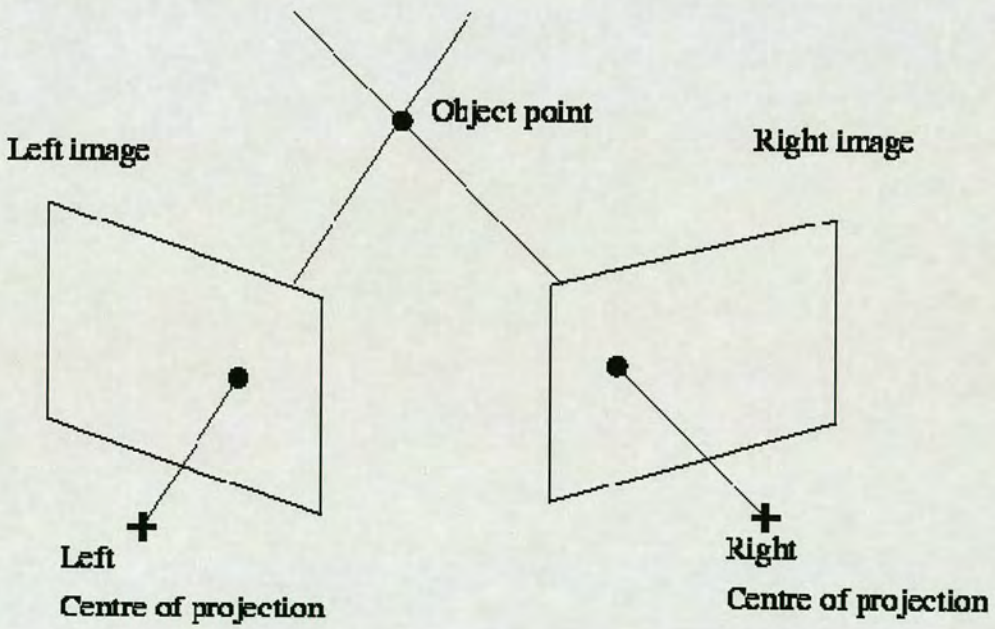


Figure 2.1: Triangulation [79]

Stereo vision, which is also called stereopsis [04], is a passive technique for extracting the 3-D structure of a scene. The 3-D structure can be only revealed if there are two views of the scene. A pair of stereo cameras is usually used to reveal the structure (depth information) of the scene. The basic principle involved in the recovery of depth using stereo vision is triangulation [40]. For example, given a pair of images taken by the stereo cameras of the same scene, the three-dimensional location of any visible object point must have their projections in both left and right images (**Figure 2.1**). For each object project in the left and right image, there is a straight line passing through itself and the centre of projection. The object point is the intersection of the two lines. Finding the intersection of these two lines is called triangulation. To determine the position of the object point in a scene, the projections of the object point in both images need to be found and matched. The process of matching the image location of the object point in one image to the location of the same object point in the other image is called correspondence.

The epipolar constraint avoids searching for the correspondence through the whole image. It reduces this search to a single line.

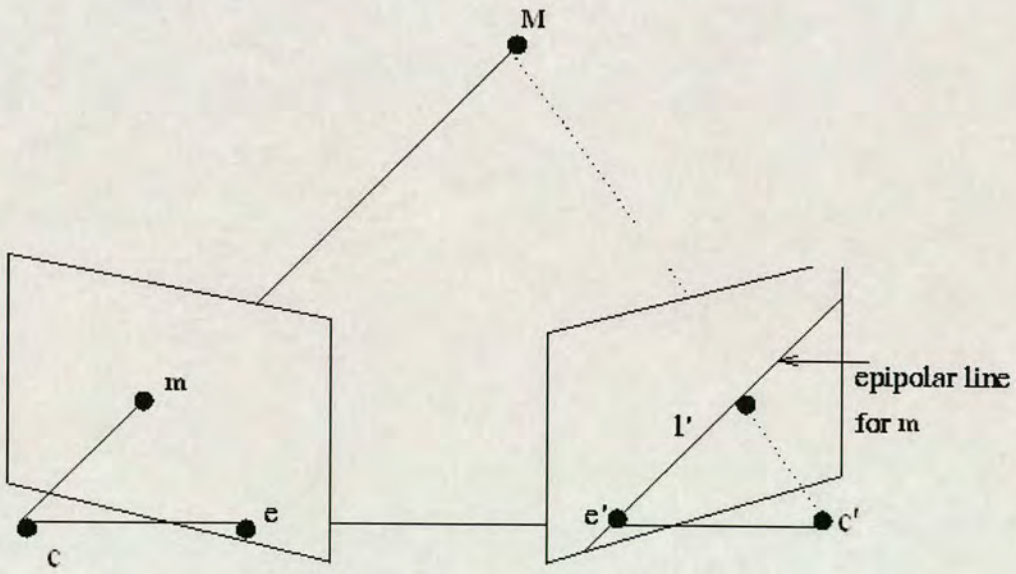


Figure 2.2: epipolar constraint [79]

Figure 2.2 shows how the epipolar constraint can reduce the correspondence search to the epipolar line. As shown in **Figure 2.2**, the object point M its projection m in the left image is lying on the straight that passes through the object point M and the left optical centre C. The epipolar plane is the plane defined by the object point M and the optical centres C and C'. The epipolar line of the left object projection in the right image is the straight line of intersection of the epipolar plane with the right image plane. This epipolar line in the right image is the projection of the part of the epipolar plane which is visible to the right camera. If the object point has its projection in the right image (visible to the right camera), its projection must lie on the epipolar line. Thus, for the example shown in **Figure 2.2**, the projection m in the left image has its epipolar line. And the projection of the object point M in the right image (the correspondence of m) must lie on the epipolar line. The search for correspondences is thus reduced from a region to a line.

Even though the complexity of the search of correspondence is reduced by the epipolar constraint, it is still a difficult job. To simplify the search process, the stereo cameras are usually placed and set up in a special way. In **Figure 2.3**, the stereo cameras are set up in such way that the two image planes are horizontally displaced and are coplanar in space, and the two cameras have identical focal length.

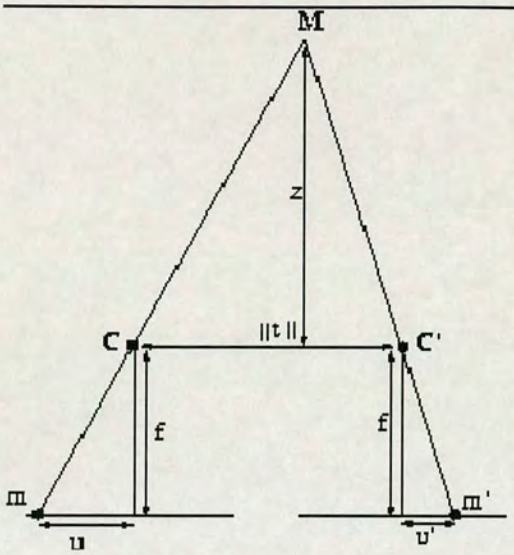


Figure 2.3: special stereo camera configuration: two image planes are coplanar, M: object point, Z: distance from object to camera baseline, C, C': optical centres, $\|t\|$: camera baseline, f: focal length, m, m': object projections in images, u, u': image coordinates of m, m'.

In this case, the disparity is defined as,

$$d = u - u', \qquad \textbf{(Equation 2.1)}$$

u' is the coordinates of m', (u',v') which is the projection of the object point M in the left image and u is the coordinates of m', (u,v) in the right image. Note that $v = v'$ as there is no vertical parallax between the two cameras. The *depth measure* z of M is related to the disparity value d as follows

$$z = \frac{f\|t\|}{d} \qquad \textbf{(Equation 2.2)}$$

In parallel camera configurations, the epipolar lines coincide with the horizontal scanlines (**Figure 2.3**). Thus, stereo matching is greatly simplified for parallel cameras.

Matching is perhaps the most important stage in stereo computation. In [04], the author divides the stereo matching into two major groups, area-based matching and feature-based matching. Area-based stereo techniques are based on intensities and their patterns: a pixel and its local neighbourhood are both considered. The pixel correspondence is established on the correspondence of intensity pattern of the pixel's local neighbourhood. While feature-based stereo techniques use symbolic features derived from intensity images rather than image intensities themselves. Hence, these systems are more stable against changes in contrast and ambient lighting. The features used most commonly are either edge points or edge segments. The disparity map is the output of the stereo correspondence. It can be generated based on either left or right image in gray scale. The intensity of each pixel is the value of the disparity of this pixel between the left and right image. The disparity map represents the 3D information of the scene taken by the stereo cameras in the stereo pair according to **Equation 2.2**.

As a basic technique regarding 3D revealing and reconstruction, stereo has been studied for over two decades. [4] also gives an overview of the major steps involved in the process of stereopsis, namely, preprocessing, stereo matching and depth reconstruction. Different techniques have been reviewed especially in stereo matching. The stereo correspondence is always a problem in stereo vision. Occlusion is always difficult to handle and the matching requires a high computational load. A dense disparity map is difficult to obtain. Area-based and edge-based matching techniques are dedicated to easing this problem.

In [5], high-accuracy stereo depth maps are generated using a special technique. They use structured light to uniquely label each pixel in a set of acquired images, so that correspondence becomes straightforward, and dense pixel-accurate correspondences can be automatically produced to act as ground-truth, except for the occluded area. On their website [6], different stereo matching algorithms are tested

with their specially produced stereo images. Although many scientists have developed many methods to solve the stereo correspondence problem, the problem is not totally solved. A dense disparity map is not easy to compute in practice. The challenge becomes how to get a sparse disparity map quickly and how to make full use of it.

Stereo vision is greatly used in the field of vehicle applications since 3-D information is demanded. A high accuracy stereo vision system for obstacle detection and vehicle environment perception is presented in [7, 8]. The disparity map is calculated by an area-based correlation algorithm. A limit is set for the correlation to lower the number of false matches. Objects are segmented in the reconstructed 3-D satellite view image. Tracking of the objects is realized by a linear Kalman filter. The object segmentation in this approach is not an easy task, since the reconstructed data is sparse. The results show vehicles are tracked on a highway. Some other objects are detected as vehicles but it is better than to mis-detect the vehicles. However, the Kalman filter is unlikely to be good enough to predict more complex vehicle motion such as turning and overtaking vehicles.

A more sophisticated way to detect objects is presented in [9]. In this approach, the urban road is not considered as planar. The road is modeled as a quasi-plane, which is a plane in general but whose normal vector is constrained within a range around the normal of the base plane. Pixels in the disparity map are classified into plane and non-plane by a RANSAC (RANdom SAmple Consensus) approach. The pixels above the plane are considered as objects. A watershed algorithm is used to segment objects. The results show that a small amount of small objects such as pedestrians and bicycles can be detected in a clear road background, but no further object classification performed.

Instead of detecting vehicles, stereo is used to detect 3D lanes in [10]. The lane is modeled as a 3D surface, defined by the vertical and horizontal clothoid curves, the

lane width and the roll angle. The detection is integrated into a tracking process though Kalman filtering. This method works well on an open road with few vehicles. Performance could be affected if the road is occupied by vehicles with a very small part showing in the image.

In [11], the view angle is greatly increased by using wide angle lenses. Using this method can avoid using multiple camera pairs or rotating cameras which causes computational load increase and decreases the robustness of the system. However, such lenses are extremely bulky, expensive and suffer from wide angle distortion.

Multiple-camera approaches, based on stereo vision, are used to reduce the occlusion problem. A wide area human tracking system using multiple cameras is described in [12]. The multiple camera system consists of several pairs of stereo cameras. Each stereo system works as an independent system to detect and track objects within its scene. Tracking results from all stereo pairs are sent to the multi-camera tracker. The multi-camera tracker is responsible for resolving local tracks from all cameras that correspond to the same object into one global track. That means that the identity of an object is carried across camera views. This system is claimed to work in real time. This system is combined with local and global tracking of objects. Object detection here is less complex since the system is mainly used in indoor environments and the majority of the objects are humans. Objects are tracked using a representation of shape, appearance and ground-based depth. All these representations together with ground-based space-time cues are used to match different trackers of an object as it moves from one camera to another. However the handling of undetected objects or new emerged objects is not clearly stated.

2.3 Camera Calibration

In the field of 3D reconstruction, camera calibration is the first and crucial step. The

aim of camera calibration is to find the parameterization of the lines through the optical centre of the camera defined by the points of the image. Many camera calibration methods exist. In [13], several methods such as Hull, Fraugeras, Tsai and Weng, are reviewed and compared. These are frequently used camera calibration methods. They are able to calculate the parameterization from one or more images of an object of known size and shape. However, when the camera calibration is involved in a visual task, the speed of camera calibration is critical. If the camera is also moving, the extrinsic parameters are changing from frame to frame. Even the intrinsic parameters such as focal length, will change if there is a zooming process in the task. This raises the problem of calibrating moving cameras.

In [14], a moving camera calibration method is described. Kruppa's equations [49] yield two algebraic constraints on the dual conic to the image of the absolute conic. Each constraint is of degree two. Thus four constraints on the dual conic can be obtained with two epipolar transformations. Since a general conic in the plane has five degrees of freedom, camera calibrations compatible with two epipolar transformations can be described with an algebraic curve. In the proposed method, three epipolar transformations are available and two dual curves are constructed such that the curves have a common singular point of order three. The camera calibration corresponds to one of the points of intersection of the two curves.

The method above, known as self-calibration, is useful for stable reconstruction of unknown environments such as outdoors. However, it requires a long sequence of images taken from unconstrained camera positions and feature matching among frames. As a result, the computational load is large. And it sometimes needs priori information about the constraint or the scene change if the camera motion is constrained or the scene changes as the camera moves.

A camera calibration method called simultaneous calibration is carried out in [15]. The idea is to place an easily distinguishable planar pattern with a known geometry

in the scene. They detect an unoccluded portion of the pattern image in each frame, compute the 3D position and zooming of the camera from it, and remove the pattern image by segmentation. They start by assuming the camera position is formed by rotating and translating it from the world origin. A homography [50] is generated by rotation, translation and zooming, and has seven degrees of freedom. Assuming a Gaussian distribution of measurement errors, an optimal estimate of the homography is obtained by maximum likelihood estimation by minimizing the average squared Mahalanobis distance of a set of feature points.

2.4 Multi-Camera Approaches

Based on the stereo theory, many multi-camera approaches exist in the literature. Comparing to stereo cameras, multi-camera approaches are able to observe a wider range of scene. For the same scene area, more occlusion problems can be solved by finding the correspondence in the extra cameras than the stereo approach. However, finding multi-camera correspondence results in heavier computational load.

In [16] three scene representations: volumetric representation, layered representation and multiple depth maps are reviewed. The volumetric representation uses a voxel concept to prevent visible surface pixels from being classified as potential disparities in the regions they occluded. Compared to conventional stereo depth maps, the volumetric approach is a much more powerful representation for dealing with partially occluded regions and mixed pixels. Unfortunately, this power comes at the expense of two problems: the depth values are quantized, which can lead to aliasing effects, and the representation has a very large number of degrees of freedom, which makes it difficult to find the optimal solution. While in a layered representation, the images are decomposed into layers, the layered approach to 3D reconstruction represents the scene as a collection of approximately planar layers. The multi-view stereo matching produces estimates for a subset of the input images, thereby representing depth in partially occluded regions and explicitly modeling the variation

in appearance between views.

The issue of using multi-cameras for road traffic monitoring is considered in [17]. The aim is to remove the classic monocular ambiguities and to retrieve the objects' height. Cameras are calibrated relative to each other using geometric constraints induced by the road pattern. The matching method relies on homographies. The road is modeled as the reference plane homography and is heightened to all heights of interest. When a point is matched by an inter-image homography corresponding to some height, the retrieval of its 3D coordinates is straightforward. The height information is directly obtained. With the inter-camera homography, each additional camera adds confidence in point positioning.

2.5 U-V-disparity

In the stereo based approaches, there have been two major challenges "facing" most of the researchers, occlusion and a dense disparity map. Occlusion has been the famous problem in stereo correspondence. It could be relieved by using multi-camera approaches, but that results in great computational load and the problem still cannot be totally solved. A dense disparity map is always desired for 3D reconstruction or object segmentation for 3D tracking. A real dense disparity map requires point to point correspondence which is greatly time consuming and almost impossible even for a median size image pair. Currently, most popular correspondence approaches include area-based and feature-based approaches. After eliminating mismatched pixels, the resulting disparity maps are from these two kinds of approaches usually sparse. The U-V-disparity approach provides a unique way of using the sparse disparity maps.

The idea of V-disparity was first introduced in [23]. Then the authors extended this idea into V-disparity and U-disparity in [24]. The "V-disparity" concept is aimed at

simplifying the process of separating obstacles from road surfaces. It is mainly used to extract surface planes which are perpendicular to the vertical plane of the stereo rig, like road surface planes and vehicle back surfaces. "U-V-disparity" in [24] relates the representations of different surfaces in U-disparity and V-disparity. It is able to classify the 3D road scene into relative surface planes and characterize the features of road pavement surfaces, roadside structures, and obstacles.

[23] explains the construction of the "V-disparity" map. Supposing that a disparity image d has been computed for the stereo image pair, let H be the function of the image variable d such that

$$H(d) = d_v \qquad \textbf{(Equation 2.3)}$$

d_v is the so called "v-disparity" image. H accumulates the points with the same disparity that occur on a given image row i. For the image row i, the abscissa u_M of a point M in d_v corresponds to the disparity Δ_M and its grey level i_M to the number of points with the same disparity Δ_M on the line i:

$$i_M = \sum_{P \in d} \delta_{vp,i} \delta_{\Delta p, \Delta M}, \qquad \textbf{(Equation 2.4)}$$

Where $\delta_{i,j}$ denotes the Kronecher delta. Once the disparity map has been computed, the V-disparity map is built by accumulating the pixels of the same disparity value in the disparity map along the i axis.

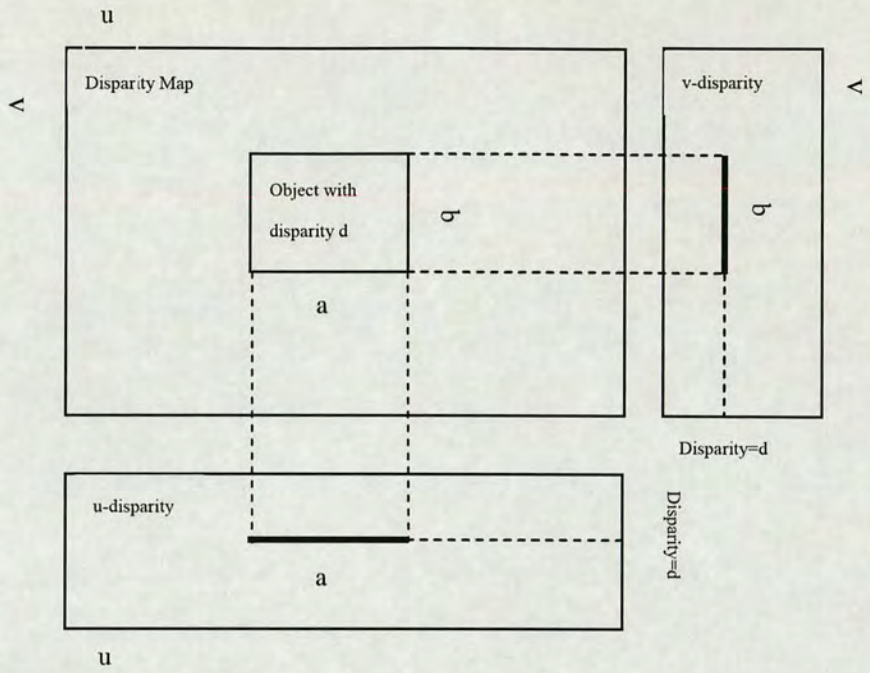


Figure 2.4: illustration of U-disparity and V-disparity map

[23] is among the first papers to propose the idea of V-disparity for this type of application. V-disparity concentrates the disparity pixels onto a horizontal direction according to the disparity value. The calculated V-disparity map provides a good representation of the geometric content of the road scene. Sparseness and error are reduced and segmentation could be performed directly on the V-disparity map since vehicles and the road surface became special lines in the V-disparity map.

The V-disparity idea is extended in [24] and a U-disparity map is used to reveal more geometric content of the road scene. It claims different types of planes in the 3D world are visible either in the V-disparity or U-disparity map. With both a U-disparity and V-disparity map, they could reconstruct the 3D world. However, this is only possible for an indoor scene with simple and regular planes. The correspondence of lines in the U-disparity and V-disparity maps is important for that purpose.

V-disparity is used for object detection for off-road vehicle navigation in [25]. This is very challenging since the off-road scenario produces more irregular shapes. A V-disparity map is used to estimate the camera's pitch oscillation caused by the vehicle movement. The results are not too satisfactory but it shows the possibility of using the V-disparity to model a non-planar surface.

Good examples of using a V-disparity map to detect and segment vehicles on a road with only a sparse disparity map are presented in [26, 27]. [28] uses a V-disparity map to detect vehicles as well. However, it also provides a way to use a V-disparity map to improve the original disparity map. Furthermore, it tries to discriminate different vehicles directly in the V-disparity map.

In [18], the authors try to generate an appearance based virtual view using images taken by multiple-cameras of a "3D room". They are able to generate images from any virtual view point between two selected real views. The virtual view generation method is based on the image-based rendering framework for temporally-varying events taken by multiple camera images of the "3D room". In this method, the virtual appearance view-points are specified by the relative position of the view points of the input image. It is not controlled by the explicit 3D points but rather by location relative to the input cameras.

2.6 Markov Random Fields (MRF)

U-V-disparity provides a unique way of using the sparse disparity map, especially for object segmentation and detection which is important in a tracking system. In this area, Markov Random Fields (MRF) theory is also a powerful approach.

Markov Random Field theory provides a convenient and consistent way of modelling context dependent entities such as image pixels and other spatially correlated

features. It has been widely employed to solve vision problems at all levels. This section mainly discusses the use of MRFs in low level vision problems, such as motion detection and segmentation.

The definitions of MRF (Markov Random Fields) are as following [55]: Let F be a family of random variables defined on the set S, in which each random variable Fi takes a value gi in L (label). F is called a random field. F is said to be a Markov Random Field on S with respect to a neighbourhood system Ne if and only if the following two conditions are satisfied:

$$P(g) > 0, \forall g \in F$$ Positivity **(Equation 2.5)**

$$P(g_i \mid g_{s-\{i\}}) = P(g_i \mid g_{Ne_i})$$ Markovianity **(Equation 2.6)**

where $S-\{i\}$ is the set difference, $g_{s-\{i\}}$ denotes the set of labels at the sites in $S-\{i\}$ and $g_{Ni} = \{g_{i'} \mid i' \in Ne_i\}$ stands for the set of labels at the sites neighbouring *i*.

A set of random variables F is said to be a Gibbs random field (GRF) on S with respect to Ne if and only if its configurations obey a Gibbs distribution. A Gibbs distribution takes the following form:

$$P(g) = Q^{-1} \times e^{-\frac{1}{T}U(g)}$$ **(Equation 2.7)**

where $Q = \sum_{f \in F} e^{-\frac{1}{T}U(f)}$ **(Equation 2.8)**

is a normalizing constant called the partition function, T is a constant called the temperature which shall be assumed to be 1 unless otherwise stated, and $U(g)$ is the energy function. The energy $U(g) = \sum_{cq \in C} V_c(g)$ is a sum of clique potentials

$V_c(g)$ over all possible cliques C. the value of $V_c(g)$ depends on the local configuration on the clique *cq*.

An MRF is characterized by its local property (the Markovianity) while a GRF is characterized by its global property (the Gibbs distribution). These two types of properties are equivalent. F is an MRF on S with respect to N if and only if F is a GRF on S with respect to *Ne*. The particular value of the MRF-GRF equivalence is that it provides a simple way of specifying the joint probability. The joint probability $P(g = F)$ can be calculated by specifying the clique potential functions $V_c(g)$ and choose appropriate potential functions for desired system behaviour. In this way, the a priori knowledge or preference about interactions between labels can be generated.

MRF models the system propagation probability as a Gibbs distribution by defining potential functions. A Gibbs distribution is more general than a Gaussian distribution which makes it able to fuse more information of the system into the distribution.

In [19], the motion estimation and segmentation scheme is based on two complementary estimating models: a classical gradient-based method, leading to estimates of quality on dense sets, and a moving edge estimator, enabling motion measurement on motion and intensity discontinuities. The two different methods are integrated with a MRF-MAP (Maximum A Priori) estimating scheme. To incorporate the two models with their different constraints, the image is divided into two parts, "smooth" and "discontinuous". Since the gradient-based method gives result at smooth areas and the edge estimator gives information at discontinuous areas. The results from the two parts are combined together with the MRF-MAP criterion to find the global minimum. The results are not very good but they are better than those performed separately by these two methods.

The method of using MRF is determined by choosing the potential function. In the

above paper [19] MRF is used to fuse two different method together by combining the different constraints, while in [20] [21], two MRF models are used for the purpose of reaching the global minimum faster.

[20] presents a two-pass algorithm for estimation of motion vectors from image sequences. The motion estimation is formulated as a problem of obtaining the maximum a posteriori in the Markov random fields (MAP-MRF). The mean field theory (MFT) is specially chosen to conduct the MAP search. MFT assumes that the effect on one particle imposed by its neighbouring particles is approximated by an average magnetic field formed by these particles. Thus in the MRF neighbourhood system, the intensity of each site in the system is approximated by an average intensity so that the whole neighbourhood system can be considered as one first order clique. In the application of motion estimation, the intensity constancy is assumed. In the two pass algorithm, the sites are categorized into three cases, "unpredicted", "uncertain" and "predicted" according to their intensity changes between different frames. Two MRFs are used, a motion vector field for "uncertain" and "predicted", and an unpredictable field for "unpredicted". The "unpredicted" category contains all discontinuities in the image, which is taken care of by the truncation function instead of using another MRF. The experimental results in the paper show that results gained from the two-pass algorithm are better than those from both block matching algorithm and a single MRF. There are two specialties in this paper: the usage of MFT and the truncation function. The MFT has the promise of cutting a nice tradeoff between efficiency and effectiveness, approximating the globe minimum by using an average value instead of each single value. The usage of a truncation function reduces the number of MRFs. They both reduce the computational complexity and give a better balance between efficiency and effectiveness.

In [21], two MRF models are also defined. One is the spatial energy function and the other is the temporal energy function. The spatial energy function uses the

smoothness a priori in the current image to divide it into several regions. It is combined with the region growing algorithm; this generates the boundaries between different regions. After the regions are generated, the temporal energy function can extract the objects from the background based on regions by calculating change of numbers of pixel in each region within successive frames in a sequence. By performing the combination of these two MRF models, this method can deal with new emerging objects.

A novel hierarchical Markov random field model to segment moving objects from image sequences is proposed in [22]. The uniqueness in this paper is of motion classification and choosing of Markov random fields. First the image is divided into several small regions by a watershed algorithm. In the motion classification, the author uses background as dominant motion; any other motion which violates the dominant motion will be marked as foreground. The weight given to each pixel is deduced from the M-estimator and represents the likelihood of background. Two MRF models are defined to capture different visual features on variant scale partitions of the frame. One MRF model labels the object region by employing motion and colour features on the large-scale partition. This MRF model provides initial segmentation based on the hybrid criteria of motion and colour. The other MRF model defined on the small-scale partition refines the object mask boundary by colour and smoothness characteristics. This is a hierarchical MRF model, the process of detection and refining makes the results quite satisfactory.

MRF is a powerful tool for segmentation of still image and motion. Object and motion segmentation are both needed in tracking system. Object segmentation is needed in the initialization step while motion segmentation could indicate the motion model which is a key to the performance of the tracking. However, I do not think that an MRF approach is suitable for my project. First, the tracking is demanded on a real time basis and the disparity calculation is performed on edge points with block

matching so that the disparity map is sparse. That increases the difficulty of the MRF segmentation. An alternative way is to perform MRF segmentation in both left and right images and find the correspondence for segmented objects, but this brings up the correspondence problem again. MRF is not suitable for the motion segmentation, either. Since the cameras are mounted on a moving vehicle, it will be a huge challenge to segment in such motion fields. The MRF method also brings a great computational load.

2.7 Condensation/Particle Filtering

The condensation algorithm is one kind of Monte Carlo method. The term "condensation" was coined by Isard and Blake in [29]. The condensation algorithm uses factored sampling, previous applied to the interpretation of static images, in which the probability distribution of possible interpretation is represented by a randomly generated set. Condensation uses learned dynamical models, together with visual observations, to propagate the random set over time. And the algorithm is claimed to run in real time.

The Condensation algorithm is based on Bayes' rule [52]. The first step is to find an object parameterised as x with prior $p(x)$, using observation data z from a single image. All the information regarding object x is represented by $p(x|z)$. $p(x|z)$ can be obtained by Bayes' rule:

$$p(x|z) = kp(z|x)p(x) \qquad \textbf{(Equation 2.9)}$$

k is a normalisation constant and is independent of x. However, in many visual tracking applications, $p(x|z)$ usually does not have a closed form, since the observation z tends to suggest multiple, competing hypotheses for x, as illustrated in **Figure 2.5**:

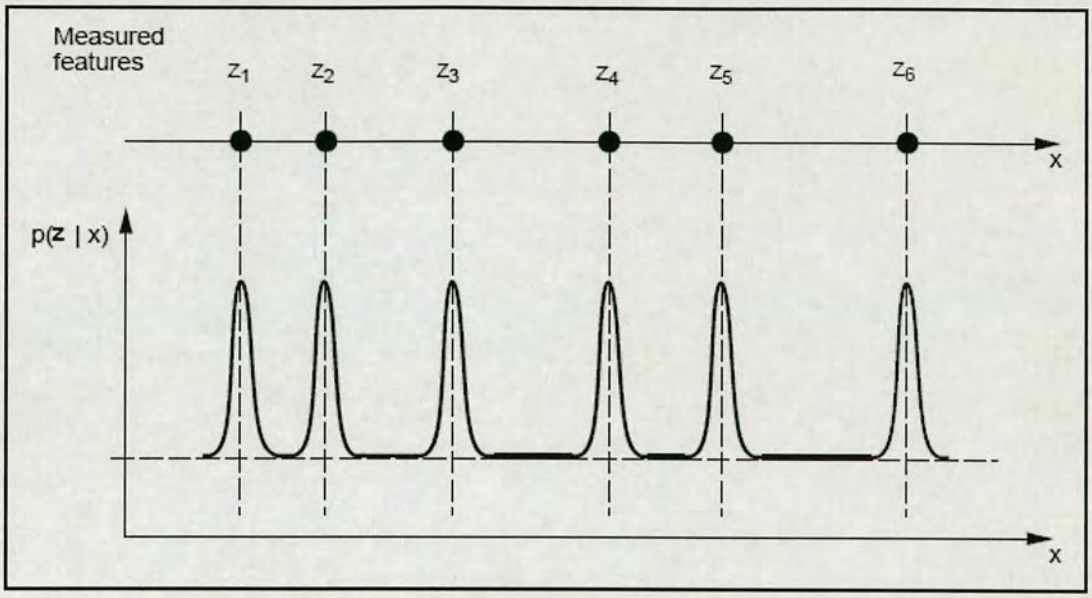


Figure 2.5: An example of one dimensional observation model [29]

Aiming at this situation, the factored sampling [53] generates a random variate x from a distribution $\widetilde{p}(x)$ that approximated the posterior $p(x|z)$. First a sample-set $\{s^{(1)},...,s^{(n)}\}$ is generated from the prior density $p(x)$ and then a sample $x = x_i$, $i \in \{1,...,N\}$ is chosen with probability:

$$\pi_i = \frac{p(z \mid x = s^{(i)})}{\sum_{j=1}^{N} p(z \mid x = s^{(j)})} \qquad \textbf{(Equation 2.10)}$$

The Condensation algorithm is based on both Bayes' rule and factored sampling, and can applied iteratively to successive images in a sequence [29]. At each time step, the output of the Condensation algorithm is a set of weighted samples $s_t^{(n)}, \pi_t^{(n)}$ which represent the current object status x. This is used as the prior for the next time step, as shown in **Figure 2.6**.

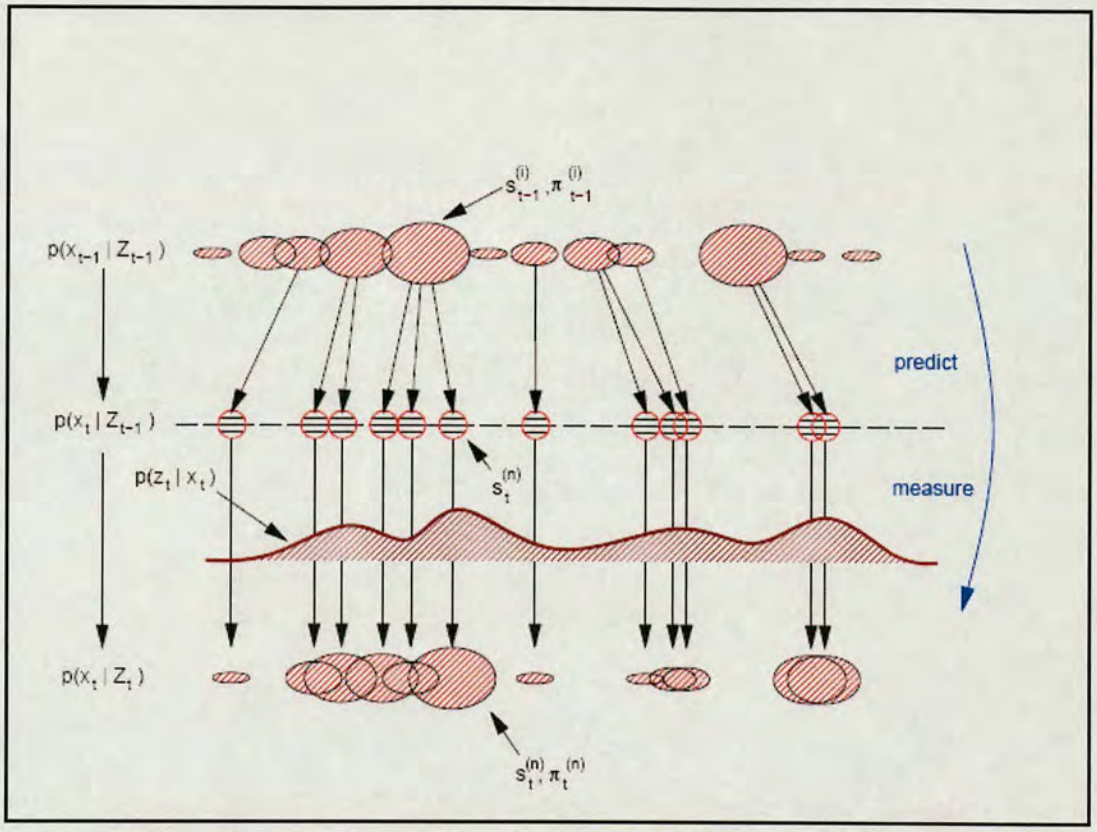


Figure 2.6: Illustration of one time step in the Condensation algorithm [29]

The condensation algorithm has been shown to offer improvements in performance over Kalman Filter [54] based approaches in non-linear non-Gaussian environment. The Kalman Filter is known as the optimal approach for sequential Bayesian filters. With the restrictions of linear and Gaussian probability distributions, no other approaches work better than Kalman Filter. However, the situation in tracking is often far from linear and Gaussian. The extended Kalman Filter addresses this problem. However, this is a sub-optimal approach, in which the probability is approximated with a Gaussian and the system and observation (which are not linear in this case) are approximated by first or second order Taylor series [44]. The extended Kalman Filter brings the Kalman Filter into a more general environment but it increases the computational expense. Compared to these Kalman Filters, the Condensation algorithm deals with a non-linear and non-Gaussian environment. It

approximates the probability distribution by Monte Carlo integration of particles instead of linearizing the problem. With enough particles, it can approximate the probability distribution perfectly. It also allows the information from different measurement sources to be fused in a principled manner. [45]

The implementation of the Condensation algorithm for this project is discussed in the next chapter.

The Particle filtering / Condensation algorithm is a powerful technique in the area of object tracking. It is a recursive Bayesian filter based on sample sets which are assigned weights able to represent the probability distribution. It also provides robust frameworks as it is neither limited to linear systems nor requires the noise to be Gaussian. Particle filtering comes from the Condensation algorithm, which was first implemented in Isard and Blake's paper [29].

In [29], the condensation algorithm is used to track curves in clutter despite elements in the background clutter which may mimic parts of foreground features. The system model is formed by feature points of the curve. And the dynamics are modeled by second order linear differences. The system can track curves in clutter in video sequences with low degrees of freedom of the curve. But the problem is that the condensation algorithm uses factored sampling, in which samples are drawn from prior $p(X)$ (which is $p(Xt|Xt\text{-}1)$), and weights are generated from $p(Z|X)$. Although it is less computational complex than particle filtering, when $p(Z|X)$ gets narrow, the factored sampling is inefficient which is known as the degeneracy problem in the condensation algorithm. When the degeneracy problem happens, the number of useful particles is not sufficient to represent the actual probability distribution in each time step. In other words, the weights assigned to the particles are small. It will eventually lead to mistracking or loss of tracking. To avoid this problem, particle filtering adds resampling in the sampling process. When the number of useful particles is under a certain limit, the algorithm resamples, to get a new set of

particles.

After the Condensation algorithm was successfully implemented in curve finding, it was also used to tracking multiple objects in [30]. The condensation algorithm is extended to track multiple objects and cope with newly appearing objects. A single condensation tracker is used for multiple objects. The probability distribution represents several objects' states simultaneously as if several individual trackers are used but assigned with different weights. To cope with newly appearing objects, a new posteriori density is generated by points from both posteriori and initialization density (directly from measurement at every time step) so that the new posteriori contains information from previous steps as well as the new status of the system. The extended algorithm is simple and can be performed in real-time, but it is not very accurate since the states of all objects are described by a single distribution.

However, Particle filtering which is derived from the Condensation algorithm, is slightly different. In Particle filtering, importance sampling is used instead of factored sampling and a resampling process is performed in order to avoid the problem of degeneracy. Particle filtering became very popular in tracking, especially in the area of multi-target tracking. In [31], both a Kalman filter and particle filter are used to track multiple objects. Rao-Blackwellisation is used to split the system model into linear and non-linear parts so that the Kalman filter and particle filter can operate on these respectively. This saves a lot of computational cost for the system since it expends the majority of the computational effort on the hard part of the problem rather than the easy part. In tracking multiple targets, computational cost is saved by using one particle filter to represent each target's distribution rather than using several filters to represent each distribution. The method works quite well on structured images, but the track initialization should be quite good and the number of targets should be known.

In [32], particle filtering is used to deal with both target tracking and sensor

management. In the first part, the target dynamics are modeled by a Jump Markov System (JMS) or Markov Switching State Space Models which accounts for the time-varying number of targets. The description of the method is quite mathematical in this paper. The basic idea of multiple target tracking is to model the system as a whole part instead of modeling each single target. In the special target model, the numbers of targets, measurements and clutters are combined together. This allows the particle filter to propagate all the dynamics from state to state. This method takes full advantage of particle filtering. It fuses all the information into a single target model. There are no results for this method in the paper, but it is quite a special way to model the system.

One trend in particle filter tracking is that colour information is used as it is good way to track non-rigid objects. In [33], objects are tracked by their colour distributions. The importance function of the particle filter is formed by the location, motion, scale and scale changing of the target area. The weights of the particles are assigned by the Bhattacharyya distance [34, 35] between the target distribution and the hypothesis distribution. In every time step the particle filter keeps the hypothesis distribution with a high similarity to the target distribution and discards the ones with low similarity. This approach tracks objects by their colour, which makes it able to track some non-rigid objects. However, object colour varies with illumination, visual angle and camera parameters. Frames where the object is occluded or too noisy must be discarded. There may be problems when two objects have very similar colour distributions, for example, in car tracking, two cars of the same make and colour.

A new approach to tracking using colour based particle filters is introduced in [36]. The tracked object is characterized by a colour probability distribution. The goal of the tracking is to find a B-spline 2D curve in the current image, such that the distribution of the interior region of the curve most closely matches the target model distribution. In particular, a Kalman particle filter (KPF) is used instead of a normal

particle filter. For each particle, a separate Kalman filter is used to generate and propagate a Gaussian proposal distribution (which is the importance function for the particle filter). Each particle evolves as a Gaussian process with its own mean and covariance; therefore, each particle may be regarded as a component Gaussian of the posterior density, whose distribution evolves according to the Kalman filter.

One of the reasons why particle filtering is so popular is that it is flexible, it can hold and pass any information which is useful. It doesn't have a defined format, and can be easily modified to suit any specific situation. A multi-dimensional particle filter is generated and described in [37]. In their paper three kinds of data; colour, motion and sound are fused together within a single particle filter. In this particular particle filter, layered sampling is performed. First, three evolution models need to be built in a similar form but independent to each other. In their tracking system, sound cues only give information about horizontal position of the sound source and it is intermittent. Motion and colour cues give information about x, y positions and the scale factor a. But they all use a Bhattacharyya similarity coefficient to define the similarity between the reference and target region. In the layered sampling process, first, particles (horizontal position Xn) are drawn from the sound evolution model and get their associate weights first (since it only gives the horizontal position). Then keep Xn and draw particles from the motion evolution model with vertical position Yn, but the weights are updated with both Xn and Yn with the full motion model. In the last step, keep Xn and Yn and draw particles from the colour evolution model with scale factor a, and update the weights with full colour evolution model. So all the information Xn, Yn and a are gained step by step. It is a novel way to do layered sampling. They do not search directly in the three dimensional state-space, but rather partition the inference in a search in one dimensional space and finish in three steps. This increases the efficiency of the particle filter and achieves the same accuracy for a smaller number of particles. However, this kind of layered sampling only works when the evolution models are independent from each other. In this paper, they fully

develop the advantage of particle filtering: combining and fusing information from different sources, and make the tracking system more robust than using the single information individually.

In [38], the authors proposed a special particle filter with joint estimation. They use the particle filter to estimate the parameters of the process as well as the state of the tracking object. Colour is chosen as the object model since it gives useful information for non-rigid objects. The observation model is a rectangle around the object. The rectangle is defined by

$$X = [x, y, w, h, xx, yy, px, py, pw, ph] \qquad \textbf{Equation 2.11}$$

in which (x, y) is the position of the rectangle (centroid). (w, h) represents the width and height respectively. (xx, yy) represents the velocities of the object. And px, py, pw, ph are four parameters of x, y, w, h, used to assume that the position, width and height of the object change with constant velocity. Finally the Bhattacharyya coefficient is calculated to measure the similarity of the histogram inside the rectangular from step to step. The results from this method show that it performs better than the conventional particle filter. This method differs from the conventional method by adding four parameters for the bounding box with the assumption that the movement and size of the box change by a constant value. The method did give good results with the test data (a van moving in a parking place), since the dynamics of the object does follow the assumption. However, the results will not be as good if the object is moving with a more complex trajectory.

A similar method is proposed in [39]. But in this, more general parameters are added into the state estimation. Once the observation model is defined, for example speed and acceleration, four basic dynamics, then a four by four transition probability matrix with sixteen parameters is added. In each step of the particle filtering, both importance function and parameters are updated by comparing $p(Xk|Zk)$ with

$p(Xk\text{-}1|Zk\text{-}1)$. This makes tracking more robust but costs more computing time.

Particle filtering/Condensation algorithms have been shown to be better than a Kalman filter when dealing with non-Gaussian, non-linear systems. However, sometimes it is still not good enough. In the situation of multiple target tracking, the computational complexity increases exponentially with the number of targets. Random movement of targets also affects the performance of particle filtering due to lack of proper priori information, which is also known as the problem of degeneracy. Occlusion is still an obstacle in tracking and particle filtering does not work well in that area, either. Thus, to solve these problems, fusing u-v-disparity and colour condensation tracking can be considered to achieve better tracking.

2.8 Conclusion

This chapter reviewed some existing literature which is related to this project. This project is aiming at on-road vehicle detection and tracking which includes disparity calculation, vehicle detection and tracking in the disparity map. The sequence used here is taken by a pair of stereo cameras mounted on a moving vehicle. Area and feature based correspondence methods are combined to calculate the disparity. To improve the accuracy, colour information is also incorporated in the matching process. The vehicle segmentation is based on the u-v-disparity method rather than the MRF segmentation due to the nature of the scene (planar feature, e.g. road and vehicle surface) and smaller computation load. Then the tracking is handled by a modified Condensation algorithm. A detailed description of the method will be introduced in later chapters.

3 Preparatory Work

3.1 Introduction

This chapter mainly aims at some preparation work for the project. As the direction of the project was clarified in the previous chapters, which is to build up a system by using stereo methods to detect and track on road vehicles, some existing projects and code were studied. Also some modification and practice were performed following this direction. Disparity calculation/stereo correspondence is studied in the first section with why the block sum of absolute difference correspondence algorithm was adopted. Camera calibration is introduced in the second section. Some lab work was done to test the camera calibration algorithm which was used in the project. After the disparity map was calculated, some segmentation algorithms are discussed, Markov Random Fields segmentation and u-v-disparity segmentation are described in section 3.4 and section 3.5 respectively. Finally, conclusions are given in section 3.6.

3.2 Disparity Calculation/Stereo Correspondence

When the research started, attention was initially paid to other existing vehicle tracking projects, such as the ECOVISION [59] project. The goal of this project was defined at quite a high level: "to design a system controlled by cognitive context which operates in video-real time and will provide the first entry point to a new generation of artifacts with a more human-like pattern of sensory-motor control. This is intended to be achieved by employing basic goal oriented cognitive properties to design and build a new generation of flexible, self-adapting artificial vision systems which operate in a real-world environment in video real-time, utilizing dedicated digital VLSI hardware for time-consuming processing steps." [59] ECOVISION was a European jointly developed project involving many universities and industrial companies. The reason why this project was initially investigated is that it was planned to be a complete artificial vision system aiming at real-time applications such as intelligent vehicles; it uses stereo image sequences but with an early cognitive mechanism. All these aspects were considered to offer valuable inputs to the project described in this thesis.

This major European Union (EU) funded project produced many progress reports [60], The project was divided into five work packages (WP). WP3-5 are most relevant to this thesis. WP3 was concerned with the space (Orientation, Stereo) and time (Motion, Flow) and their interaction. WP4 was concerned with the aspect of how to adapt these to a changing context (by remapping images). WP5 was concerned with the relevant aspects of a visual scene (with respect to a given task).

In WP3, a stereo correspondence approach was introduced. In this ECOVISION project, stereo correspondence is considered at a higher level than in other more conventional approaches. The correspondence is based on features such as colour, orientation and phase which are integrated into "primitives". Motion and stereo correspondence are made between these primitives by combining them together as support and evidence to each other. The total structure of this process is shown in **Figure 3.1**.

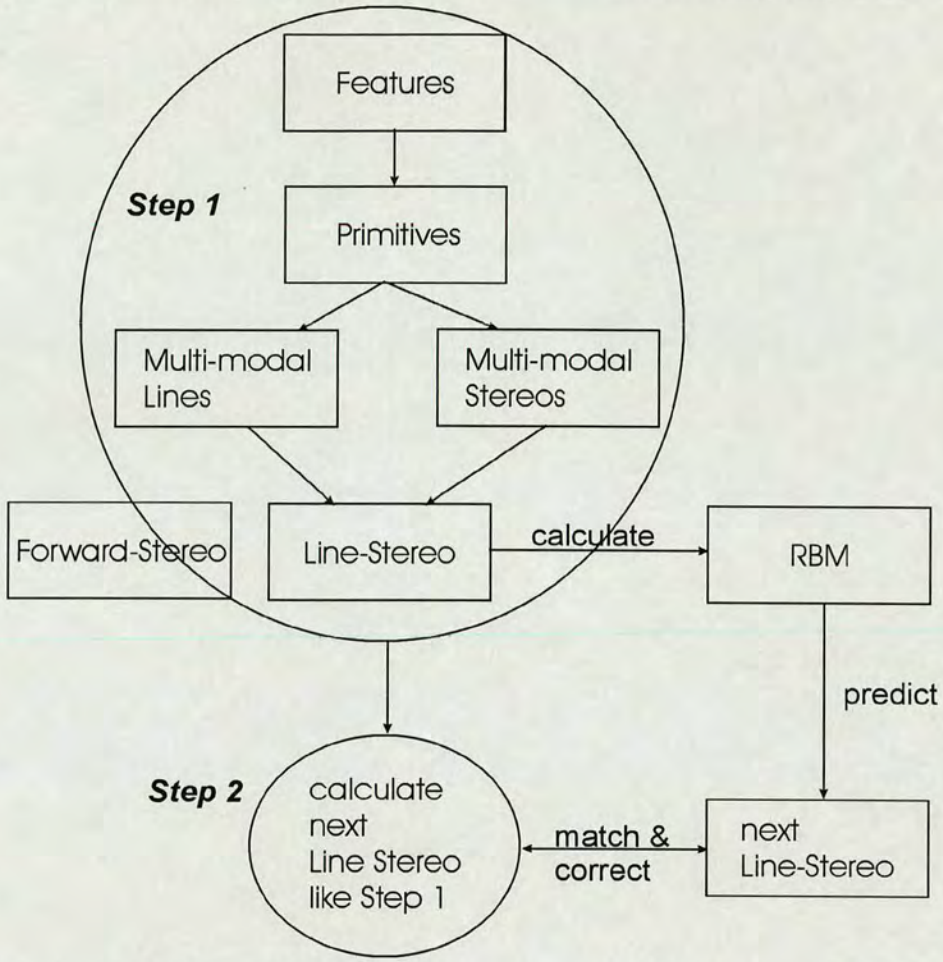


Figure 3.1: Stereo Matching Process in ECOVISION Project [60]

As noted above, in the stereo matching process, a multimodal similarity measurement was used. In [61], a compact coding of image information was used which explicitly separates geometric information (orientation) and structural information (phase and colour), and the combination of these factors was claimed to give better results on stereo matching. A monogenic signal [62] method was used to split the descriptions of geometric and structural information of grey level images. The split descriptions were used to form the stereo similarity measure function. The monogenic signal is a 2D analytic signal which is a combination of a 2D signal with a Riesz transformed one [63]. The monogenic signal performs a split of identity. It orthogonally divides the 2D signal into energetic information (indicating the likelihood of the presence of structure), its orientation and it structure (phase). The features in the image are extracted at energy maxima in local image patches (usually

where edges appear, and also the phase and orientation of the edge). Phase can be used to express the contrast transition on the edge. By adding the colour information, an original primitive includes position, orientation, phase and colour. In [64], there is a further development of the multi-modal matching function, motion vectors from optical flow are also added into the primitives as a support for the stereo matching in stereo sequences. Experimental results show that the similarity function with colour and optical flow information gives less false matching in multi-modal stereo.

As indicated in [60], with multi-modal stereo, although the relative sparseness of the primitives reduces the stereo matching problem, the definition and the 3D reconstruction accuracy would be expected to decrease accordingly. The chance for primitives to be extracted in both images from exactly the same 3D point is very low. So for the reconstruction stage, a point-line correspondence is used. The position of one primitive of the first image is matched in the second image with a whole line which is the multi-modal line. According to [65], after the features are extracted, it is assumed that there should be a set of consistent primitives of any particular structure of the scene in the image. Consistency between two primitives is defined by two criteria: collinearity and modality consistency. The collinearity criterion consists of two factors: proximity and continuation. The former evaluates how, given the position of the two primitives, a link between them is likely to exist. The closer they are, the closer the second has to be to the line defined by the orientation of the first. The continuation criterion in terms of orientation can be defined as a minimal curve joining two primitives. The modality continuity criterion over the colour and phase modalities is calculated using similarity functions for phase and colour which are described in [61,64].

In line-stereo, multi-model lines are matched considering local primitives whose information is consistent over their neighbourhood. The BSCE (Basic Stereo Consistency Event) criterion is used to measure the confidence between two potential stereo matches. The idea of BSCE is to find the line correspondence based on point-wise stereo correspondences. Lines are found by integrating matched primitives according to their's and their neighbours' attributes such as position, phase, orientation and colour. If all the matched primitives on two lines are confirmed, than

the two lines are confirmed. When the BSCE criterion is applied in the whole neighbourhood, a non-local line-stereo stereo similarity is formed.

Forward-Stereo, was invented during the third year of the ECOVISION project. There is a problem when using the stereo matching above: the reconstruction becomes unstable when the orientation of the edge is too similar to the epipolar line. It is claimed in the report [60] that the epipolar geometry in forward motion leads to a radial set of epipolar lines, intersecting at the heading point. This, as opposed to the traditional horizontal epipolar lines of the standard stereo, allows disambiguation of these horizontal structures in most cases.

The stereo correspondence process described in the ECOVISION project is a feature based process which introduces various support features into the stereo correspondence. As mentioned above, in the multi-model stereo, the matching of primitives is realized by using a monogenic signal. Thus it was decided to investigate evaluate and compare some stereo correspondence methods such as monogenic signal matching and sum of difference block matching.

3.2.1 Monogenic Signal Based Stereo Correspondence

The evaluation of monogenic signal matching was performed with the code supplied by Peter Kovesi [66]. The monogenic signal is able to perform a split of identify. The monogenic signal is an analytic signal generated by Riezs transform. It is based on irrotational and solenoidal vectors. The local amplitude and phase of the monogenic signal orthogonally represent different kinds of energetic information which indicates the likelihood of the presence of a structure, such as orientation and phase. The available code is written in Matlab [67]. It generates putative matches between previously detected feature points in two images by looking for points that have minimal differences in monogenic phase data within windows surrounding each point. Only points that correlate most strongly with each other in both directions are returned. A Harris corner detector [68] first finds the corner points in the image as feature points. Then the features such as intensity and phase (feature points found are usually edge points, phase represents the degree of intensity change from one side to the other of edge) of these features points are calculated and stored in a matrix called

the "correlation matrix". Thus, for a stereo pair, there are a left correlation matrix and a right correlation matrix. For any feature point in the left correlation matrix, its potential match in the right correlation matrix is the most strongly correlated feature points in the right correlation matrix. This correlation is performed for both directions, from left to right and right to left. Only the potential matches which are consistent in both directions are the final matches.

This monogenic signal matching code is available with correlation matching code together for comparison. In the correlation matching algorithm, putative matches between previously detected feature points are found in two images by looking for points that are maximally correlated with each other within windows surrounding each point. Only points that correlate most strongly with each other in both directions are returned. This correlation matching algorithm considers intensity of the feature points and its neighbourhood pixels.

Both algorithms were tested on images from ECOVISION project. The results are shown in **Figure 3.2-Figure 3.4**. Both algorithms were tested on two consecutive frames first. As the results in **Figure 3.2** and **3.3** for correlation and monogenic signal respectively show, all matched feature points from two frames were plotted in the first frame with a blue line connected. The results of monogenic signal matching look more like the optical flow from two frames since the images were taken by a pair of stereo cameras mounted on a moving vehicle. The monogenic signal matching definitely gives better results than correlation matching since it brings phase information into the matching process, but there are still many false matches.

Figure 3.2: putative matching of feature points in two consecutive frames by correlation: The blue lines connect the set of two matched feature points in two consecutive frames. Thus the orientation of the blue lines should look like optical flow vectors. Some false matches can be identified (for example in the red box, some blue lines are much longer than the others. They should be about the same length.)

Figure 3.3: putative matching of feature points in two consecutive frames by monogenic signal: Compared to Figure 3.2, feature points are better matched since the blue lines look more like optical flow vectors. For example in the red box, the length and orientation of the blue lines are quite consistent.

Figure 3.4: putative matches of feature points in a pair of stereo images by monogenic signal (search along epipolar line): All the blue lines connecting two matched feature points are horizontal (along the epipolar line).

Then the test of monogenic signal matching was performed on a pair of stereo images. To simplify the matching process, the code was modified and the search was limited to epipolar lines. Results are shown in **Figure 3.4**. Matched feature points from both stereo images are plotted in the left image with a green line connected. It was very difficult to judge if this was a good stereo match or not since the cameras were not totally calibrated (epipolar lines are horizontal but disparity is non-zero at an infinite distance). However, some points about monogenic signal matching can still be concluded from these results. The speed of the algorithm greatly depends on the number of feature points to be matched since it is a point-to-point matching process. Limiting the number of feature points could mean that for some feature points, their true correspondences are not detected as feature points in the other image. This results in a very sparse disparity map with many errors. In the ECOVISION project, errors could be reduced since many other features such as orientation, colour etc were used in the matching process. However, this level of sparseness is clearly unacceptable for the project described in this thesis.

3.2.2 Sum of Block Based Absolute Difference

Neither a full nor highly accurate disparity map was demanded in this project. However, it has to contain sufficient correctly matched feature points and should run much faster. The focus was therefore transferred to some more conventional stereo correspondence approaches such as Sum of Absolute Difference (SAD) as it was claimed that this has been implemented in hardware [69] and could run in real time. Following the ideas used in ECOVISION project, colour information is added into the SAD algorithm. Feature points were extracted by a Sobel filter, as basically these are just edge points. This prevented areas which contain no texture being fed into the matching process, which lowers the numbers of false matches. This Sobel filtering was performed on all three channels of the colour images. The block of the region to be matched then became a cubic of region, which contained 3 channels of Sobel filtered regions. The overall algorithm is shown in **Figure 3.6**.

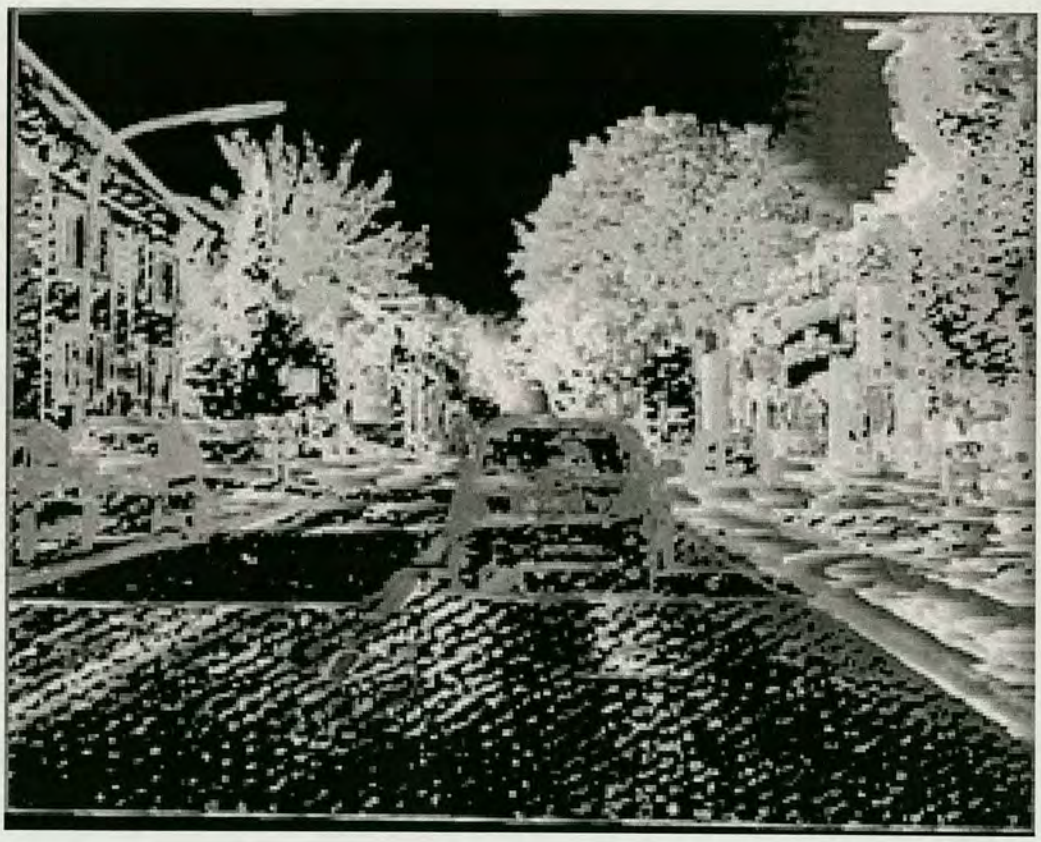


Figure 3.5: matching by block based sum of absolute difference on Sobel filtered stereo images

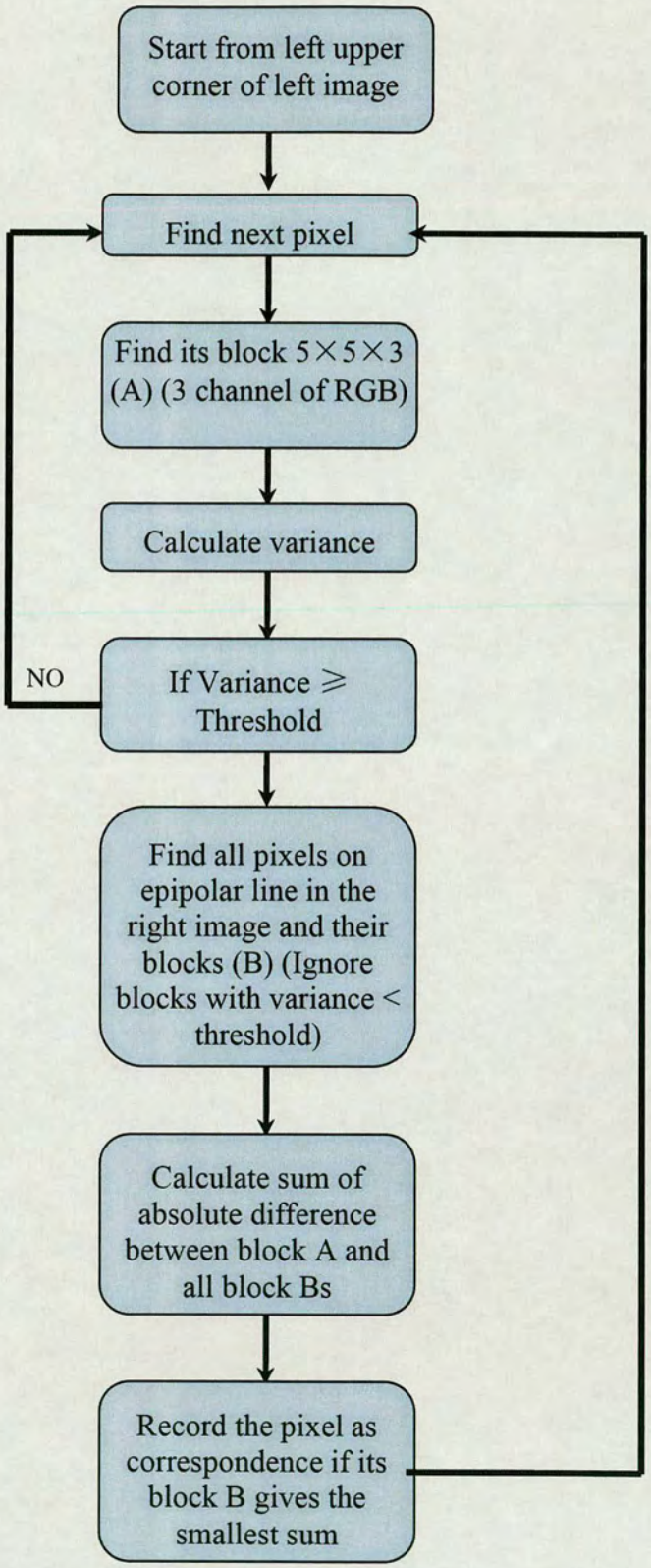


Figure 3.6: flow chart of the stereo matching algorithm

The block based sum of absolute difference matching algorithm was first tested on the same ECOVISION images. As shown in **Figure 3.5,** many more feature points were matched than with the monogenic signal. These matched points are significant enough to be grouped and segmented based on the appearance of the disparity map. Again, it was very hard to tell the accuracy of the algorithm by just looking at the disparity map since the stereo cameras were not completely calibrated. Some calibrated stereo images with camera parameters were therefore needed to further test this algorithm. This piece of work was done in the Vision Lab of the University, using the same camera calibration tool which was used in the DIPLODOC project using a pair of IceRobotics [70] stereo cameras. This will be described in the following section.

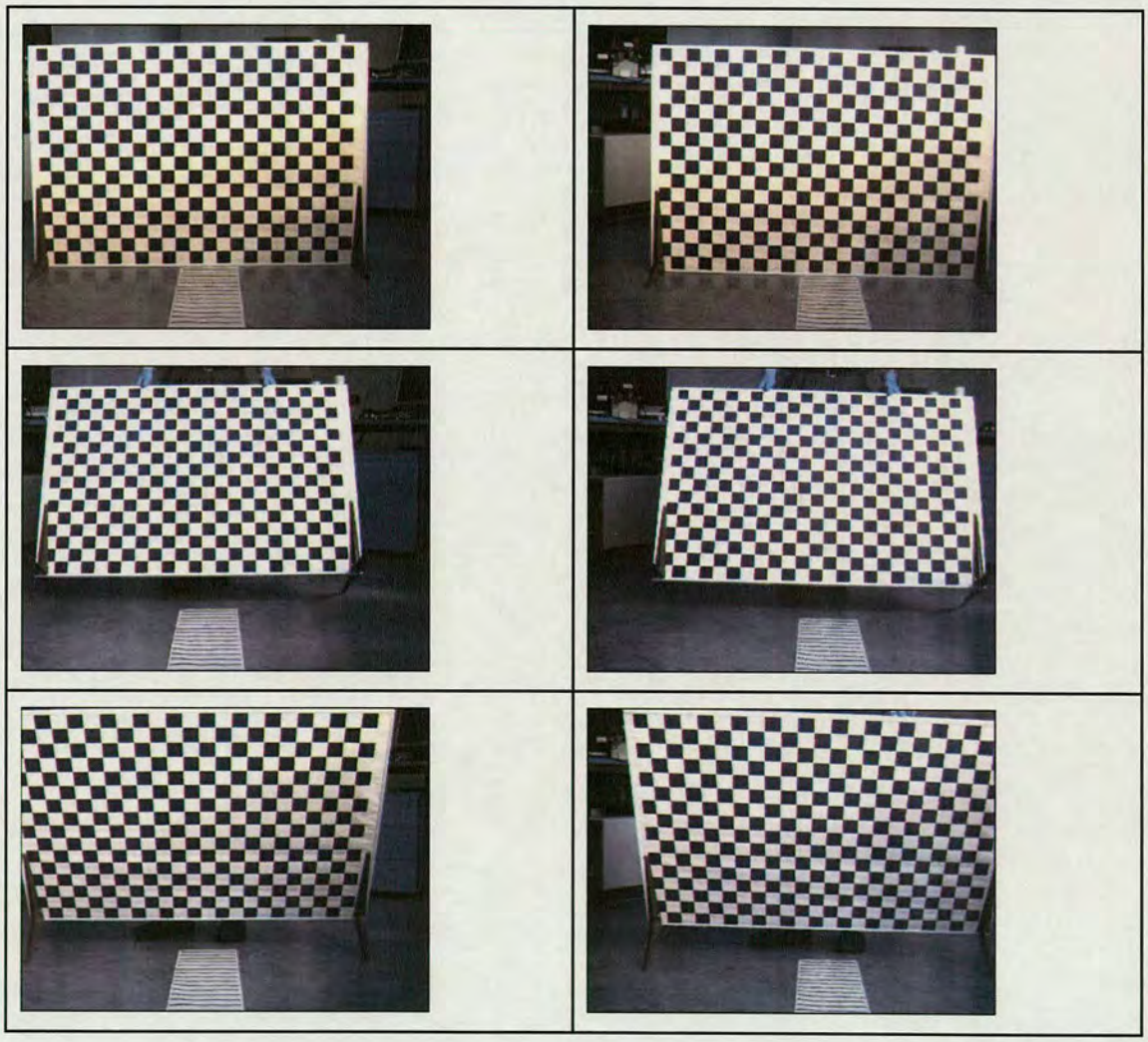
3.3 Camera Calibration

As mentioned in Chapter 2, DIPLODOC stereo sequences were used in this project. These were calibrated stereo images which were taken on a moving vehicle. For the DPLODOC images, the camera calibration tool used was DLR CalDe and DLR CalLab [71], which was based on Matlab and free for download. The work described in this section was to create the calibration stereo images to calibrate the stereo cameras in the Vision Lab and use stereo images with ground truth to verify the camera calibration tool box and the block based sum of absolute difference matching algorithm.

This calibration algorithm was a plane-based camera calibration using a series of images of a check board taken from different angles. Calibration images were first loaded by the tool and corners were extracted. The corner extraction algorithm simply made a guess of the positions of the corners according to the input numbers of squares and a hand-drawn area. This initial guess was usually good unless there was a great lens distortion. In that case, an initial guess of the lens distortion needed to be used to adjust the corner positions. After the corners were extracted, the camera calibration algorithm was done in two steps: first initialization, and then nonlinear optimization. The initialization step computes a closed-form solution for the

calibration parameters based not including any lens distortion. The non-linear optimization step minimizes the total re-projection error (in the least squares sense) over all the calibration parameters (9 DOF for intrinsic: focal, principal point, distortion coefficients, and 6*20 DOF extrinsic => 129 parameters). The optimization is done by iterative gradient descent with an explicit (closed-form) computation of the Jacobian matrix.

The examples of the camera calibration work done in the Vision lab are shown in **Figure 3.7** and **3.8**.



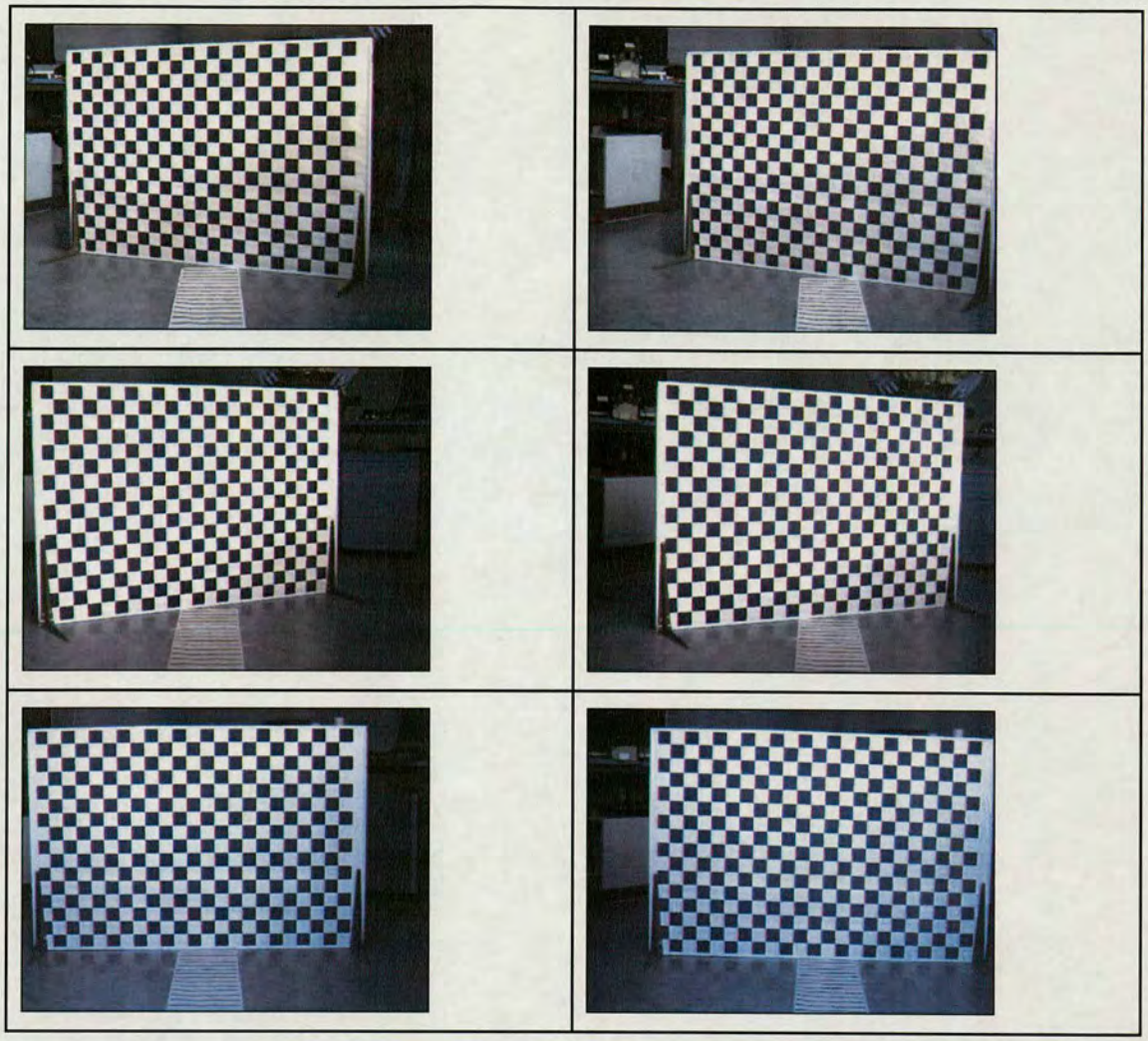
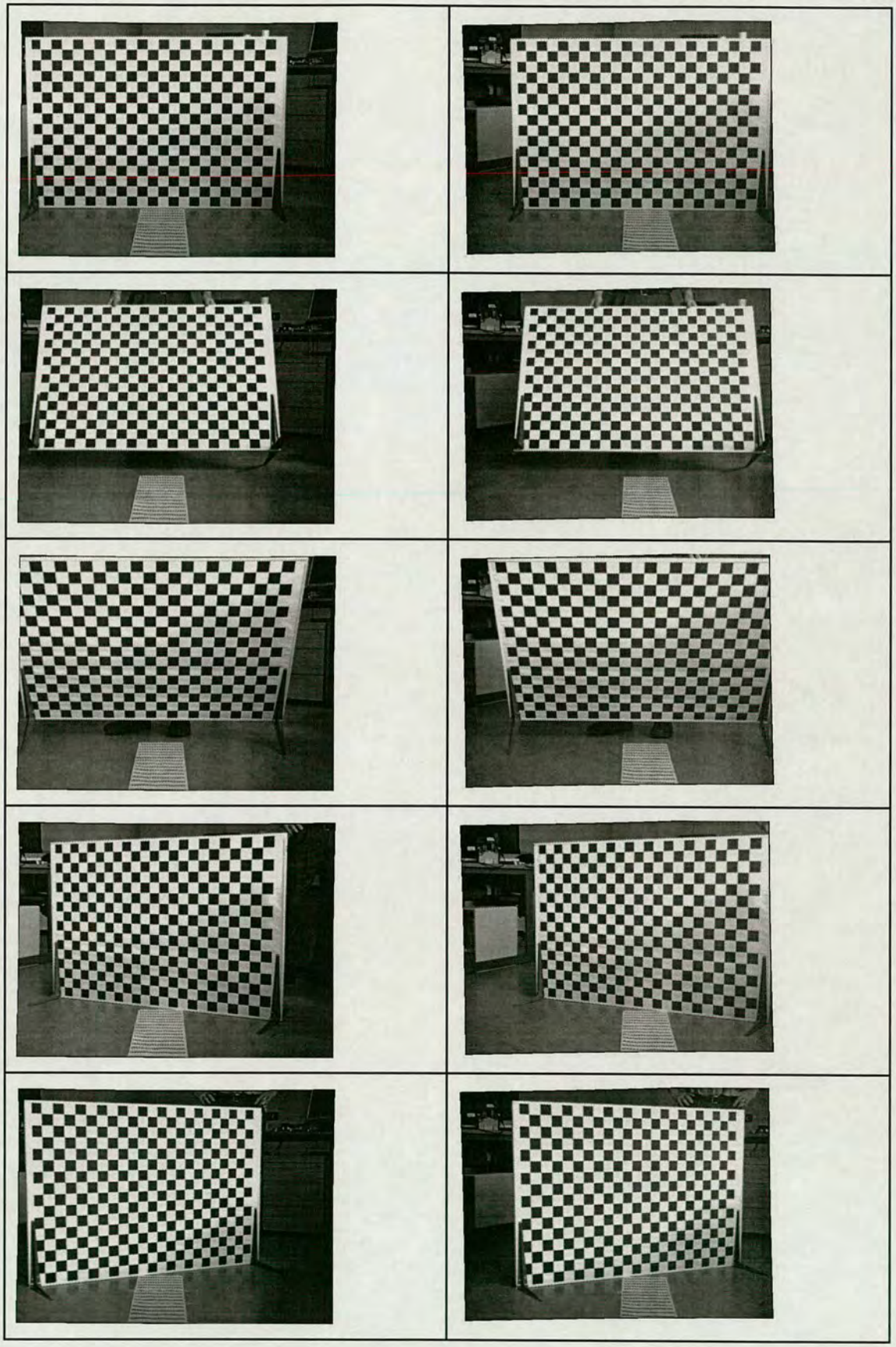


Figure 3.7: camera calibration stereo images taken in the Vision lab, Checkerboard with 60mm squares.



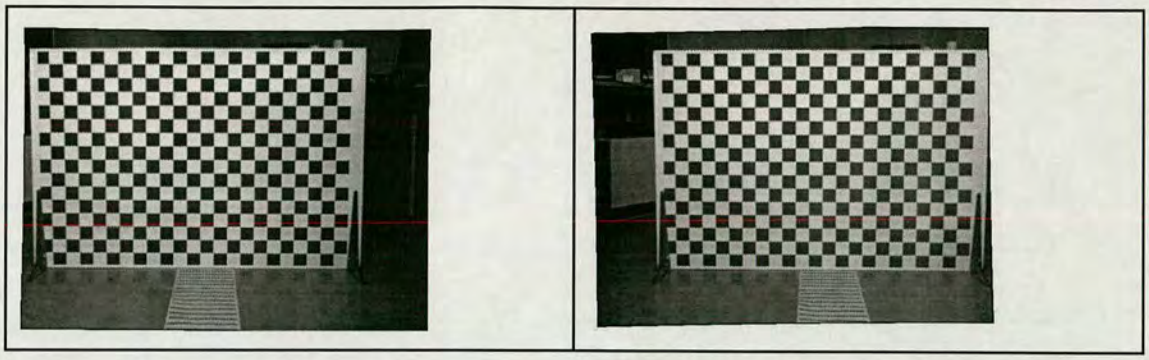


Figure 3.8: rectified images after camera calibration: all correspondent points in the left and right images have the same vertical coordinates.

The camera calibration method was derived from [71]:

The *focal length* in pixels is stored in the 2x1 vector *f*.

The *principal point* coordinates are stored in the 2x1 vector *c*.

The *skew coefficient* defining the angle between the *u* and *v* pixel axes is stored in the scalar α.

The *image distortion coefficients* (radial and tangential distortions) are stored in the 5x1 vector *k*.

Let M be a point in space of coordinate vector $X = [X_c;Y_c;Z_c]$ in the camera reference frame. M is firstly projected onto the image plane according to the intrinsic parameters (f, c, α, k). Let m_n be the normalized (pinhole) image projection:

$$m_n = \begin{bmatrix} X_c / Z_c \\ Y_c / Z_c \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} \qquad \textbf{(Equation 3.1)}$$

Let $r^2 = u_n^{\,2} + v_n^{\,2}$ **(Equation 3.2)**

Considering lens distortion, the new normalized point coordinate x_d is defined as follows:

$$m_d = \begin{bmatrix} u_d \\ v_d \end{bmatrix} = \left(1 + k(1)r^2 + k(2)r^4 + k(5)r^6\right) + dm \qquad \textbf{(Equation 3.3)}$$

where *dm* is a tangential distortion vector:

$$dm = \begin{bmatrix} 2k(3)u_n v_n + k(4)(r^2 + 2u_n^2) \\ k(3)(r^2 + 2v_n^2) + 2k(4)u_n v_n \end{bmatrix} \quad \text{(Equation 3.4)}$$

The final pixel coordinates *x_pixel* = *[u;v]* of the projection of *P* on the image plane are:

$$u = f(1)(u_d + \alpha v_d) + c(1) \quad \text{(Equation 3.5)}$$

$$v = f(2)v_d + c(2) \quad \text{(Equation 3.6)}$$

Therefore, the pixel coordinate vector *[u;v]* and the normalized (distorted) coordinate vector x_d are related to each other through the linear equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} \quad \text{(Equation 3.7)}$$

where *K* is known as the camera matrix, and defined as follows:

$$K = \begin{bmatrix} f(1) & \alpha f(1) & c(1) \\ 0 & f(2) & c(2) \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(Equation 3.8)}$$

This camera calibration tool was used to calculate all the distortion parameters according to the calibration images. With the images shown in **Figure 3.1**, the parameters calculated for both left and right cameras are shown in **Table 3.1**:

Left camera	Right camera
f=[1158.835; 1157.511]	*f*=[1180.744; 1178.955]
c=[311.814; 265.335]	*c*=[392.365; 255.681]
α=0.000	α=0.000
k=[-0.213; 0.332; -0.002; 0.001; 0.000]	*k*=[-0.212; 0.380; -0.001; 0.001; 0.000]

Table 3.1: calibration parameters for left and right stereo cameras

The rectified images according these parameters are shown in **Figure 3.8**. These images were rectified onto horizontal epipolar lines. For any pair of corresponding pixels $[u_l;v_l]$ and $[u_r;v_r]$, $u_r = u_l + d$, $v_l = v_r$, where d is the disparity of the stereo pair.

After the calibration was done, some extra images were taken with known object distances (see **Figure 3.9**). The disparity maps were calculated by the sum of absolute difference algorithm. The camera calibration verification work here was to calculate the object distance from the calculated disparity value and compare this to the true distance.

Left rectified	Right rectified
 120 cm	 120 cm
 140 cm	 140 cm

Figure 3.9: real rectified images using calibration parameters.

Since the disparity maps were calculated from rectified images, all the lens distortion parameters should be zero. Thus, for both left and right camera, $k = 0$; $a = 0$. So from **Equation 3.3**

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} \qquad \textbf{(Equation 3.9)}$$

And from **Equation 3.5 & 3.9**

$$u_l = f_l(1)u_{dl} + c_{l(1)} = f_l \frac{X_{cl}}{Z_{cl}} + c_l(1) \quad \textbf{(Equation 3.10)}$$

$$u_r = f_r(1)u_{dr} + c_{r(1)} = f_r \frac{X_{cr}}{Z_{cr}} + c_r(1) \quad \textbf{(Equation 3.11)}$$

According to the displacement of two cameras (horizontally placed, baseline length $\|t\|$),

$$X_{cr} = X_{cl} + \|t\| \quad \textbf{(Equation 3.12)}$$

$$Z_{cl} = Z_{cr} \quad \textbf{(Equation 3.13)}$$

Let the disparity of the object in the images be d, so

$$u_r = u_l + d \quad \textbf{(Equation 3.14)}$$

From **Equation 3.10~3.14,** we can get:

$$Z = Z_{cl} = Z_{cr} = \frac{\|t\|}{\frac{u_l + d - c_r(1)}{f_r(1)} - \frac{u_l - c_l(1)}{f_l(1)}} \quad \textbf{(Equation 3.15)}$$

Calculated object distances using **Equation 3.15** and true object distances of 5 pairs of stereo images are listed in **Table 3.2**

Calculated	True	Error/True
123.3 cm	120 cm	2.75%
135.6 cm	140 cm	3.14%
177.6 cm	180 cm	1.33%
223.1 cm	220 cm	1.40%
263. 5 cm	265 cm	0.57%

Table 3.2: summary of calculated and true distance

The calculated object distances are quite close to the true distances. The errors are probably due to the rounding of disparity values from true disparity values. This could be improved by using higher resolution cameras. Basically, the camera calibration tool did quite a good job here (The percentage distance error is less than 4%).

3.4 Disparity Map Segmentation

With the disparity map calculated by sum of absolute difference, work was done to examine various segmentation algorithms on this kind of disparity map. The minimum requirement of the segmentation was to at least detect the vehicle region in the disparity map.

As described in Chapter 2, Markov Random Fields (MRF) are popular in image segmentation. To get a better understanding of MRF, a simple sample code of an MRF algorithm [72] was downloaded to test on the disparity maps. In this MRF algorithm, the energy function is formed by adding a singleton and doubleton together. The singleton follows the Gaussian probability density function, it takes the form of: σ

$$En_{Singleton} = \log\left(\sqrt{2\pi\sigma} + \frac{p_i - \mu^2}{2\sigma}\right), \qquad \textbf{(Equation 3.16)}$$

Where p_i is the chosen pixel; σ and μ are the variance and mean calculated from the defined neighbourhood (e.g. 4-pixel neighbourhood). The singleton represented the possibility of whether a certain pixel belongs to a defined class just on its own.

The doubleton takes the following form:

$$En_{Doubleton} = \beta \cdot \gamma(p_i, p_j), \qquad \textbf{(Equation 3.17)}$$

$$\text{Where } \gamma(s_i, s_j) = \begin{cases} -1, & s_i = s_j \\ 1, & s_i \neq s_j \end{cases} \qquad \textbf{(Equation 3.18)}$$

Where p_i is the chosen pixel and p_j is any pixel in p_i's neighbourhood; β is the scale which measures the significance of the doubleton. The doubleton considered the energy relationship of the pixel and its neighbourhood pixels. In this algorithm, it only considered its first order neighbours, 4 neighbourhood pixels, looking at the intensities: if they had same intensity value, it was negative while positive for different intensity values. So that the global energy to minimize is:

$$En = \sum_{i,j} En_{Singleton} + En_{Doubleton} \qquad \textbf{(Equation 3.19)}$$

There are four different optimization algorithms to find the global minimum of the energy function: Metropolis, Modified Metropolis Dynamics (MMD) [73], Gibbs Sampler [74] and Iterated Conditional Modes (ICM) [75, 76].

Among these optimization algorithms, Metropolis, MMD and Gibbs sampler are simulated annealing methods. They are essentially methods for a partially random search of the solution space. At each step of the algorithm, the previous solution is subjected to a random perturbation. They don't always follow the decrease of the energy function; an increase of the energy is also allowed according to a certain criterion. This is because an uphill move is sometimes necessary in order to prevent the solution from settling in a local minimum. The probability of accepting uphill moves is controlled by a temperature parameter. They start with a high temperature and almost all random values are accepted. Then the temperature starts to cool down and the system reaches a steady state. ICM is a deterministic algorithm which aims to reduce the computational load of the simulated annealing methods. It could be posed as a special case of both Metropolis and Gibbs sampling algorithms. As this method is a supervised one, number of classes and their training sets should be chosen at the beginning of the program. Then average μ and variance σ of each are calculated for each class. **Figure 3.10** shows the test image used in this algorithm ("chess board" image with 4 classes, and SNR = 3dB noise added [77].

Figure 3.10: The original image, with -3dB noise added

The detailed algorithms of the four methods and their test results are shown in **Tables 3.3-3.6**.

- **Metropolis**

In the Metropolis algorithm, at each step a new candidate solution (classification mark) is generated at random. If this new solution decreases the energy function, it is always accepted; otherwise, it is accepted according to an exponential probability distribution, $P=exp(-\Delta E/T)$, where ΔE is the change in the criterion due to the perturbation, and T is the temperature parameter. If T is relatively large, the probability of accepting a positive energy change is higher than when T is small for a given ΔE.

Algorithm (parameters needed: temperature T, local energy change threshold α, and global energy change threshold t):	Result
1. Calculate all Singletons and for each site using μ and σ from all classes. Find the minimum local energy E_local (singleton, old) and mark the site with the according class. 2. Calculate global energy E_global (all pixels) according to the initial classification from step 1. 3. Set the global energy change E_change = 0. 4. For each site, give a random fluctuation to the class mark (new). Calculate new local energy E_local(singleton+doubleton, new). If (E_local(new)–E_local(old))/T>=logα*, accept the new class mark, E_change=E_change+|E_local(new)–E_local(old)|. Else, keep the old class mark. 5. Decrease T as T=T*C(C<1) 6. If E_change>t, go to step 3. Else, stop. *: 0<α<1.	 Metropolis, global energy=48061.2, iteration=212, cpu time=4860ms

Table 3.3: Metropolis algorithm and it test result

- **MMD**

In the modified Metropolis algorithm the only difference is that instead of using α as threshold of the accepting probability $P=exp(-\Delta E/T)$, a random number r $(0<r<1)$ is chosen as the threshold.

Algorithm (parameters needed: temperature T, and global energy change threshold t):	Result
1. Calculate all Singletons for each site using μ and σ from all classes. Find the minimum local energy E_local (singleton, old) and mark the site with the according class. 1. Calculate global energy E_global (all pixels) according to the Gaussian classification from step 1. 2. Set the global energy change E_change = 0. 3. For each site, give a random fluctuation to the class mark (new). Calculate new local energy E_local(singleton+doubleton, new). If (E_local(new)–E_local(old))/T>=log(r*), accept the new class mark, E_change=E_change+|E_local(new)–E_local(old)|. Else, keep the old class mark. 4. Decrease T as T=T*C(C<1) 5. If E_change>t, go to step 3. Else, stop. *: random number r, 0<r<1.	 MMD, global energy=48049.1, iteration=177, cpu time=4010ms

Table 3.4: MMD algorithm and its test result

- **Gibbs Sampler**

In Gibbs sampling, the perturbations are generated according to local conditional probability density functions derived from the given Gibbsian distribution *(P=exp(E_local/T)*, in this example). And the random number *r (0<r<1)* is generated to decide which classification mark to chose.

Algorithm (parameters needed: temperature T, and global energy change threshold t):	Result
1. Calculate all Singletons for each site using μ and σ from all classes. Find the minimum local energy E_local (singleton, old) and mark the site with the according class. 1. Calculate global energy E_global (all pixels, old) according to the Gaussian classification from step 1. 2. Set the global energy change E_change = 0. 3. for each pixel, calculate Esum_local = Σexp(-E_local(all classes)/T). Calculate E_gibbs=E_local/ Esum_local and add E_gibbs class by class until the sum is larger than r (random number, 0<r<1) and assign pixel with the according class. 4. Calculate E_global(new) according to new class mark from step 4. 5. E_change=E_change+|E_global(new)-E_global(old)|, decrease T as T=T*C(C<1), E_global(old)=E_global(new). 6. If E_change>t, go to step 3. Else, stop.	 Gibbs Sampler, global energy=48008.8, iteration=162, cpu time=7990ms

Table 3.5: Gibbs Sampler algorithm and its test result

- **ICM**

Being different from the simulated annealing algorithms, the temperature T is set to be zero for all iterations in the ICM algorithm. And the probability of accepting perturbation that increases the energy is always 0. It only accepts a decrease of the energy.

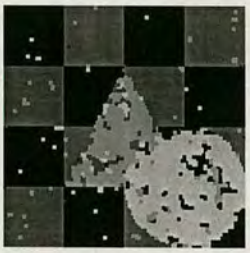
Algorithm (parameters needed: global energy change threshold t):	Result
1. Calculate all Singletons for each site using μ and σ from all classes. Find the minimum local energy E_local (singleton) and mark the site with the according class. 1. Calculate global energy E_global (all pixels, old) according to the Gaussian classification from step 1. 2. Set the global energy change E_change = 0. For each pixel, assign any new class to the pixel if only E_local(singleton+doubleton, new)>E_local(singleton+doubleton, old) 3. Calculate E_global(new) according to new class mark from step 3. 4. E_change=E_change+|E_global(new)-E_global(old)|, E_global(old)=E_global(new). 5. If E_change>t, go to step 3. Else, stop.	 ICM with global energy=50148.7, iteration=10, cpu time=700ms

Table 3.6: ICM algorithm and its test result

- **Discussion**

Metropolis, MMD and Gibbs sampling are all simulated annealing algorithms, they allow perturbations that increase the energy in order to prevent the solution from settling in a local minimum. However, as the candidate values are generated by random perturbations, the algorithm needs a large number of iterations for convergence. Thus the computational load of these algorithms is significant, especially when there are a large number of candidate values. Slightly different from the Metropolis, Gibbs sampling accepts the random perturbations according to a

Gibbs distribution. That makes it slightly quicker than Metropolis. ICM converges much faster than other three, as it only allows those perturbations decreasing the energy. However, it is likely to get stuck in a local minimum (which causes more false classifications then the other algorithms, as seen in **table 3.6**). Thus, it is critical to initialize ICM with a reasonably good initial estimate.

The MRF segmentation method provides a way of integrating different segmentation criteria into a global energy function. The MRF segmentation algorithm described here is a supervised one. It requires pre-defined classes with class attributes. However, in this project, the segmentation is to segment on-road vehicles from road scene background based on disparity maps. It is relatively difficult to define a vehicle class just based on disparity. And vehicles vary in shape, colour and speed, which make it difficult to construct a set of common attributes for a vehicle class.

3.5 U-V-Disparity Segmentation

In this project, the segmentation algorithm should ideally be more straight-forward and unsupervised, to detect and segment vehicles on road in the disparity map. In [23], the disparity map was interpreted into a U-disparity and V-disparity map. Following this interpretation of the disparity map, a straight-forward vehicle algorithm was developed and tested for this project.

According to [24], different planes in the real 3-D world show as straight lines in the U-V-disparity map. In the U-disparity map, planes parallel to the horizontal axis (camera baseline) show as vertical lines. Especially, when the plane is parallel to the camera plane, it shows as a vertical line. Thus, the back of a vehicle going straight ahead (with respect to the camera vehicle) shows as a vertical line while the road surface shows as a straight line with slope between $(-\infty,0)$. In the V-disparity map, planes in the vertical axis show as straight lines. As in the U-disparity map, planes parallel to the camera plane show as vertical lines. Other planes parallel to the

horizontal axis (such as side of a vehicle, buildings on the road side) show as straight lines with a non-zero slope.

On a normal road scene, only the back of the vehicles are visible, the majority of which consists of planar surfaces. These planar surfaces are usually parallel to the camera plane. These surfaces are straight lines in both V-disparity and U-disparity maps. As shown in **Figure 3.11**, an object has a planar surface which is parallel to the camera plane. Its interpretations in the U-disparity and V-disparity maps are both straight lines. And the length and position of the lines indicate the size and position of the object respectively. Some simulated and real images exercises were done following this idea for planar surface detection and segmentation in a disparity map.

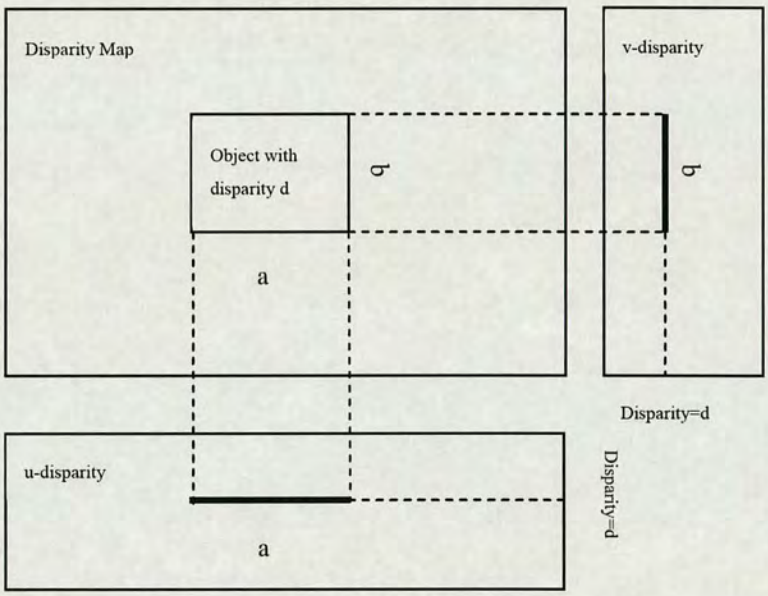
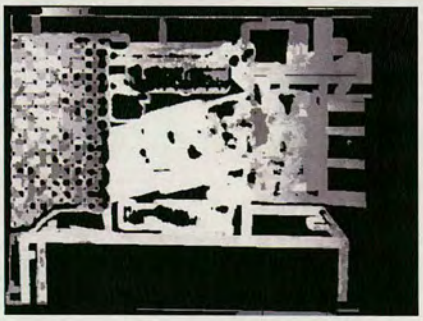


Figure 3.11: illustration of U-disparity and V-disparity map

Figure 3.12: U-V-disparity bounding result of "Teddy"

The image used in the first figure is from Middlebury stereo images [78]. This disparity map is a ground truth full disparity map. The intensities of "Teddy" in the disparity map are the same. "Teddy" can be treated as planar surface parallel to the camera plane. By finding the lengths and positions of its u-disparity and v-disparity lines, a bounding box can be drawn around "Teddy". This is a quite straight-forward detection and segmentation of "Teddy".

(a)



(b)

Figure 3.13: (a)&(b) U-V-disparity bounding result of "blue box", (a): left stereo image, (b): disparity map

Another exercise was done using the images taken in the Vision lab. In **Figure 3.13**, a bounding box was successfully drawn around the "blue box". For this real image, its disparity map was calculated by the block based sum of difference algorithm described above. The disparity map was sparse and with some mis-matchings which is common for normal segmentation algorithms.

It can be seen that U-V-disparity detection and segmentation is straightforward and robust. Although it is very difficult to accurately segment the object with a clear boundary, a bounding box showing the position and volume of the object is enough for the purpose of the project. Detailed U-V-disparity detection and segmentation algorithms will be discussed in the next Chapter.

3.6 Conclusion

This chapter has described some preparation work in the project. Some key algorithms for the initialization part of the project were chosen and developed through this preparation work. For the project, the disparity maps were calculated by a block based sum of absolute difference algorithm and vehicle detection by u-v-disparity segmentation. The detail of the final approach of the project will be discussed in the next chapter.

4 Main approach

4.1 Introduction

In this chapter, different approaches which were implemented on the test sequence are discussed. The work started with vehicle detection through U-V-disparity. Vehicles are detected in each frame independently using this method, and it is called still frame vehicle detection in this chapter and is discussed in detail in the first part. The second part looks into details of the conventional Condensation algorithm. A particular implementation of it for this project is then introduced. Details of the final approach are given in the third part. The final approach combines U-V-disparity detection and Condensation tracking together. In particular, the conventional Condensation algorithm is modified to suit this combination better.

4.2 Data Sequence

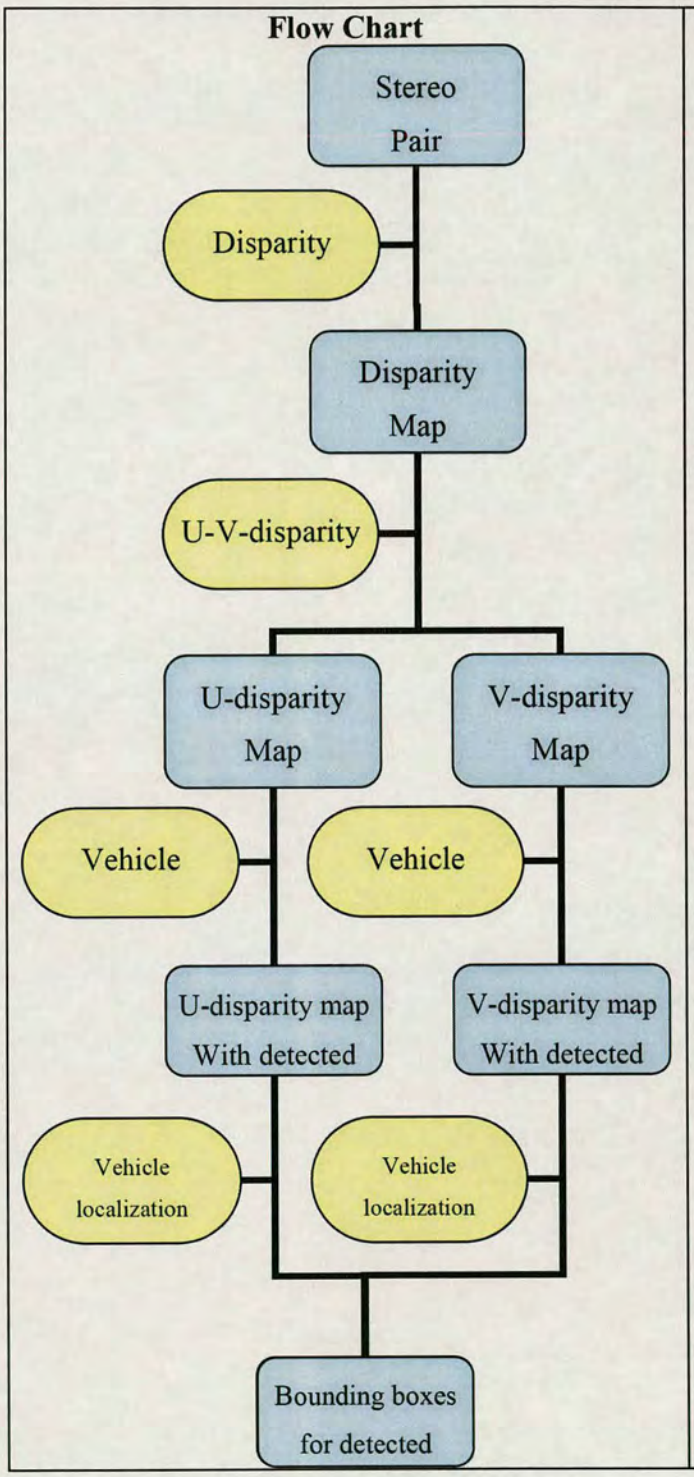
The date sequence used in this project is the DIPLODOC road stereo sequence [48]. The sequence consists of 865 image pairs taken from a stereo camera mounted on a moving vehicle. The images were captured on July, 16 2004, about 11am, near Trento, Italy. The acquisition device is a Videre Design MEGA-D stereo camera pair installed near the rearview mirror. The sequence is 15 fps, 320x240, colour. The images are saved in a lossless format (PNG) without any pre-processing. The stereo pair is calibrated with the SVS software [51]. The sequence is the composition of five subsequences, each of them presenting very different traffic and road conditions: from highway-like roads to urban scenarios with crossroads, parking lots and complex environment; from congested traffic to completely free road.

For the purpose of vehicle tracking, a section of 50 frames of this sequence in urban scenarios was used. This was because there are more vehicles appearing in this section. This part of the sequence starts with 2 vehicles in the foreground, a black car and a white van.

Figure 4.1: Selected frames from sequence: the bounding boxes in the images show the results of the tracking, which will be explained in the next chapter.

As seen in the images in **Figure 4.1**, the black car is in the nearest position while the white van is partially occluded by the black car. As the sequence proceeds, both of vehicles are turning to the right. Then, a few frames later, another van overtakes on the left side with quite a high speed. At the end of the sequence, the first van goes off the image, and the black car is partially occluded in the image.

This part of the sequence consists of several complex urban road situations. Multiple vehicles are showing in the sequence. This also includes a new vehicle emerging and an old vehicle disappearing. Vehicles showing in the sequence are of different types; a car and a van. These vehicles are with different motion types as well: going straight, turning, and overtaking. Partial occlusion also exists with these vehicles. Considering all of the above factors, this part of sequence is quite suitable for testing a multi-vehicle tracking algorithm.



Algorithm

1. calculate the disparity map for the stereo pair using a colour image sum of absolute difference block matching algorithm
2. calculate the U-disparity and V-disparity for the calculated disparity map
3. threshold the U-V-disparity map with the defined threshold, then detect vehicles in the U-V-disparity map, getting disparity value for detected vehicle
4. for detected vehicles, find the left-right boundaries and up-down boundaries in the U-disparity map and V-disparity map respectively
5. draw bounding boxes around the detected vehicles according to their boundaries

Table 4.1: Still frame U-V disparity algorithm

4.3 Still frame vehicle detection by U-V-disparity

Initial work in this area, concentrated on vehicle detection algorithms. In the special case of a stereo sequence, the U-V-disparity method was investigated. The U-V-disparity map was used to detect vehicles and find their positions. The algorithm developed for the U-V-disparity detection was for a still frame and was independent for each frame. The detection results were found to give values quite close to the real positions of the vehicles which is critical for the success of a whole vehicle tracking system. **Table 4.1** shows the detail for the still frame U-V-disparity detection algorithm which was used for this work.

4.3.1 Disparity Map Calculation

Since the stereo pair is in colour RGB format, the disparity calculation includes colour information, in other words, using all three channels of RGB. The disparity calculation is a time consuming task so a simple sum of absolute difference (SAD) block matching algorithm is used, since it can be implemented on an FPGA to allow the calculation to be done in real time [42,43]. As mentioned previously, as the resolution of the images is 320×240 which is quite small, the accuracy of a full disparity map calculation is fairly low. Errors usually occur when the algorithm tries to find the stereo correspondence for a smooth area with little texture or intensity variation. To solve this problem, the disparity is calculated on the edge points instead of the whole image. The stereo pair is filtered by a Sobel filter in all three RGB channels separately which generates a pair of "colour edge images". The three layers of the "colour edge image" are the Sobel filtered results from RGB channels of the original image. The sum of absolute difference (SAD) is calculated on the "colour edge image" pair. As, the 'block' becomes three dimensional, a $5\times5\times3$ block is used. This is big enough to include the edge change in the block while small enough to limit the expanding of edge points in the disparity map. The calculated disparity map is sparse but with less error matching than a full disparity map.

4.3.2 U-V-disparity calculation

Since the disparity map is sparse, it is difficult to segment vehicles directly from it. The *U-V*-disparity map is able to project the disparity pixels vertically or horizontally according to each disparity, which concentrates the pixels in one direction and makes segmentation one dimensional. The U-disparity and V-disparity calculation algorithms are described in **Table 4.2**.

According to [24], different planes in the real 3-D world show as straight lines in the U-V-disparity map. In the U-disparity map, planes parallel to the horizontal axis (camera baseline) show as vertical lines. Especially, when the plane is parallel to the camera plane, it shows as a vertical line. Thus, the back of a vehicle going straight ahead (with respect to the camera vehicle) shows as a vertical line while the road surface shows as a straight line with slope between $(-Inf, 0)$. In the V-disparity map, planes in the vertical axis show as straight lines. As in the U-disparity map, planes parallel to the camera plane show as vertical lines. Other planes parallel to the horizontal axis (such as side of a vehicle, buildings on the road side) show as straight lines with a non-zero slope.

U-disparity calculation algorithm	**V-disparity calculation algorithm**
1. find the size (r, c) of the calculated disparity map 2. find the disparity range $(d_{\min}, d_{\max})$ in the calculated disparity map 3. the size of the U-disparity map is $(c, d_{\max} - d_{\min})$ 4. for each pixel in the U-disparity map, the intensity $I(i, j), 1 \le i \le c, 1 \le j \le d_{\max} - d_{\min}$ equals the number of pixels in the i_{th} column of the disparity map with disparity equal to $j + d_{\min} - 1$.	1. find the size (r, c) of the calculated disparity map 2. find the disparity range $(d_{\min}, d_{\max})$ in the calculated disparity map 3. the size of the V-disparity map is $(r, d_{\max} - d_{\min})$ 4. for each pixel in the V-disparity map, the intensity $I(i, j), 1 \le i \le c, 1 \le j \le d_{\max} - d_{\min}$ equals the number of pixels in the i_{th} row of the disparity map with disparity equal to $j + d_{\min} - 1$.

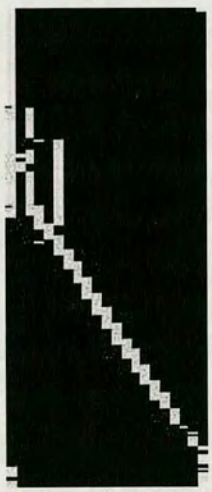
Table 4.2: Disparity calculations

4.3.3 Vehicle Detection in U-V-disparity Map

A:U-disparity map

B: V-disparity map

C: U-disparity map after thresholding



D: V-disparity map after thresholding

Figure 4.2 (A-D): U-V-disparity map

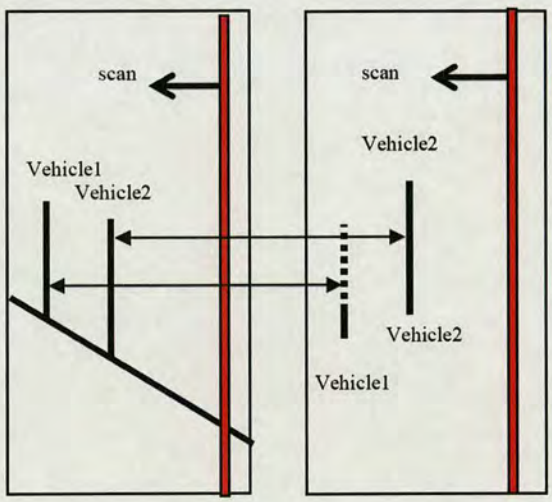
As mentioned above, the disparity map calculated from the edge stereo pair, is sparse. It is very difficult to segment vehicles directly from the disparity map. While U-V-

disparity projects the pixels in the disparity map horizontally and vertically which makes segmentation one-dimensional. The error can be reduced by properly thresholding the U-disparity and V-disparity map.

As seen in **Figures 4.2 (A-D)**, in the original U-disparity and V-disparity map, the intensity indicates the number of disparity pixels. The vehicles are the only parts of interest in the disparity map, in other words vertical lines in the U-disparity and V-disparity map. In the thresholded maps, all major lines become clearer. This is greatly dependent on a well chosen threshold. The **Figure 3.11** shows how the threshold is chosen.

Figure 3.11 shows the relationship between the disparity map and the corresponding U-disparity and V-disparity map. If a rectangular object ($length = a, height = b$) with disparity d exists in the disparity map, its correspondence line in the U-disparity is a straight line with $d = 7$ and $length = a$, the intensity of each pixel on the line $i = b$ ($length = b, i = a$ in V-disparity map). This is for a dense disparity map. If the disparity map is sparse as the case here, the intensity of the line pixel should be smaller than b in the U-disparity map. The threshold should thus be smaller than b but big enough to eliminate other 'noise' pixels and errors. The threshold is chosen to correspond to the real world size of the object. If the length of the back of the vehicle in the real world is roughly 2 metres, based on the stereo camera calibration parameters, the length of the back of the vehicle in the image is $a = 2 \cdot s / d(pixels)$. Where s is the ratio between d and real world size which is calculated by the camera calibration parameters. In this application, the ratio s is 6.7, if the disparity is 7 pixels, the length of the back of the vehicle is roughly 50 pixels in the image. Considering that the disparity map is sparse, it is assumed that half of the vehicle edge pixels are corresponded for the disparity map, so that the threshold is set as 25 pixels. If the thresholded U-V-disparity map uses a threshold of 25 pixels, the results are adequate. Thus, in the thresholded U-disparity and V-disparity map, only lines of vehicles (vertical lines) and road surface (line with slope) will remain.

After obtaining the thresholded U-disparity and V-disparity map, the vehicle detection is quite easy and straightforward. The target of the vehicle detection in this stage is to find the disparity values of the existing vehicles. The algorithm simply scans over the columns of the thresholded V-disparity map. If the number of non-zero pixels in the current searching column is over a certain limit, a vehicle is detected in that column. The diagram in **Figure 4.3** illustrates the process of vehicle detection.



Left: V-disparity **Right: U-disparity**

Figure 4.3: Correspondence of V-disparity and U-disparity map

In **Figure 4.3**, a vertical line (the red line) which covers the whole height of the V-disparity map, scans from right to left (arrow direction, from large to small disparity). At each disparity value, the number of pixels occurring on the line is calculated. If the number of pixels is over a certain limit, a vehicle is detected and the current disparity is the disparity for the detected vehicle. The limit is decided in the same way in which the threshold is set. The same process goes for the U-disparity map. The detected results from the U-disparity map should correspond to the results from the V-disparity map. If the correspondence is found and the disparity value is the same, the detected vehicle is verified. However, not all the lines detected in the V-disparity map have correspondence in the V-disparity. This usually occurs on detected lines with smaller disparity, since the vehicle is occluded or partially

occluded by the vehicle with bigger disparity, in other words, the vehicle in the front. In the situation of partially occluded vehicle (as shown in the diagram, the dashed line on the left indicates the part occluded by the other vehicle), the limit is set to a smaller value for that particular disparity value to detected the short line. In this algorithm, the limit is set at 10 pixels. If the vehicle is totally occluded, it is not able to be seen in the U-disparity map, and the detected result from the V-disparity map is not verified. It is not marked as a detected vehicle.

4.3.4 Vehicle Localization and Bounding Box Generation

The result from the previous step is the disparities of detected vehicles. As illustrated in **Figure 4.3**, the length of the line indicates the size of the vehicle. The ends of the line in the U-disparity map are the left and right boundaries of the corresponding vehicle while the ends in the V-disparity map are the top and bottom boundaries. The target for the vehicle localization algorithm is to find the ends of the lines for detected vehicles. For each detected disparity d, the algorithm searches along the corresponding column of the d in the U-disparity (or V-disparity) map and finds the first and last non-zero pixel, the row number of the first one is the left (or top) position in the image while the row number of the last one is the right (or bottom) position. In the case of a partially occluded vehicle, there is no problem in finding the top and bottom positions in the V-disparity map. While in the U-disparity map, either one of the two positions is not the right one. The algorithm still finds the two ends, and compares them with the end positions for the vehicle with bigger disparity (in front). If the end position drops in the range of $[left, right]$ of the vehicle with bigger disparity, that end is the wrong end. The algorithm discards the wrong one and keeps the right one. The algorithm estimates the occluded end by assuming the vehicle back is roughly 2 metres long, which is a pixels($a = 2 \cdot s / d(pixels)$) long in the image, then adds to (or subtracts from) the other end, if it is smaller (or bigger) and takes the result as the boundary position of the occluded side.

The last step is to draw a bounding box around the detected vehicles according to their boundary positions in the original image. This bounding box represents a

detected vehicle. It does not fit exactly to the shape of the vehicle but it is big enough to enclose the vehicle, and simple and reliable enough to be an object model in the tracking process, which will be explained in detail in the next section.

4.4 Consecutive frames vehicle detection and tracking (Condensation algorithm)

U-V-disparity is a fast and effective method to detect the vehicles while the condensation algorithm is a well known method of tracking. This project, is intended to not only detect vehicles on the road but track them as well. So the next stage is to use a condensation algorithm to track the vehicles.

Condensation vehicle tracking was initially started with sample condensation code [46]. This is a simple implementation of condensation. Simulated data is generated to model a particle moving in one dimension under the action of a first-order auto-regressive process with measurements corrupted by noise. The condensation algorithm then tracks the particle using a model of the same form. However, for the vehicle tracking application, the algorithm should be designed to track multiple objects (vehicles) in a two dimensional image. The algorithm was therefore modified to make it suitable for two-dimensional tracking. To complete the whole algorithm for vehicle tracking, an object model, a motion model and an observation model are needed. The correct choice of these is very important to the results of the tracking.

4.4.1 Object Model

Deciding on the object model is the first step in developing the tracking algorithm. The object model represents the object to be tracked. It should make the object recognizable to the tracker. In this vehicle tracking, the very simple model of a bounding box was used. Finding an exact model for the vehicles on a road is a very challenging job. Vehicles have different models with different shapes. Even the same

vehicle, when viewed at different angle, changes its shape dramatically in the image. Some of the object models [46] in the literature use 6 parameters to model the back of the vehicle. This is acceptable but even it is not capable of modelling a turning vehicle. Some use more complex models [29] with parameters which need to be learned from training sets. Since none of these are perfect, making the object model as simple as possible to reduce the computational load, but unique enough to make the object recognizable, is an attractive option.

A bounding box usually needs four parameters, position (x, y), height h and width w. The algorithm described here, only uses three parameters, position (x, y) and disparity d. Since the real size of the vehicle does not change, the size of the vehicle in the image is only related to its distance from the camera. If l is the length of a line in the image, and the line is z distance to the camera, then l and z are with related by the relationship, $l \propto \frac{1}{z}$. While the distance z and disparity d have the relationship $z \propto \frac{1}{d}$. That means the relationship between l and d is $l \propto d$. In other words $l = s \cdot d$, where the scale s is constant. So if s is known, l can be represented by d. If we can find two scale constants for the height and width of the bounding box, the bounding box can be represented by three parameters, $[x, y, d]$. This can be done in the initialization, which will be explained in the next section. This is a very simple object model but is good enough for basic vehicle tracking.

4.4.2 Observation Model

The observation model of the algorithm uses colour information. At each time step, the colour histograms in the bounding boxes from different particles are calculated. This colour histogram is calculated with 512 bits of colour. And these colour histograms are compared with the colour histogram from the previous step. To compare the two colour histograms, a similarity measure which is based on colour

distributions is required. A popular measure between two distributions $p(u)$ and $q(u)$ is the Bhattacharyya coefficient [34].

$$\rho[p,q]=\int\sqrt{p(u)\cdot q(u)}du \qquad \textbf{(Equation 4.1)}$$

For discrete densities such as colour histograms, $p=\{p^{(u)}\}_{u=1\ldots m}$ and $q=\{q^{(u)}\}_{u=1\ldots m}$, the Bhattacharyya coefficient is defined as

$$\rho[p,q]=\sum_{u=1}^{m}\sqrt{p(u)\cdot q(u)} \qquad \textbf{(Equation 4.2)}$$

For the coefficient $\rho\in[0,1]$, the larger ρ, the more similar two distributions are. $\rho=1$ means two distribution are perfect matching, they are identical.

4.4.3 Motion Model

A motion model is supposed to model the motion of the object under tracking, and predict the position of the object in the next time step. In this road sequence, the cameras are mounted on the moving car. This makes the vehicle motion in the image relative motion. Disregarding the camera vehicle motion, the motion of the vehicle being tracked is unpredictable. The motion is related to the path of the vehicle, for example, straight-going, turning and cutting-in. There may be great changes in the motion. It is therefore very difficult to choose a precise motion model. For this condensation tracker, a simple second order motion model was chosen. Assuming the motion change between two time steps is linear

$$[x_t,y_t]'-[x_{t-1},y_{t-1}]'=[A_x,A_y]'\cdot([x_{t-1},y_{t-1}]'-[x_{t-2},y_{t-2}]')+w_t \qquad \textbf{(Equation 4.3)}$$

Thus

$$[x_t,y_t]'=[x_{t-1},y_{t-1}]'+[A_x,A_y]'\cdot([x_{t-1},y_{t-1}]'-[x_{t-2},y_{t-2}]')+w_t \qquad \textbf{(Equation 4.4)}$$

Where x_t and y_t are the position for time step t. $[A_x, A_y]'$ are scales for the position change. w_t is a noise term. The motion model here only uses x and y, without d, since d is determined in each time step during vehicle detection.

This motion model is able to model constant acceleration motion. However, by tuning $[A_x, A_y]'$ in each time step, it is able to model acceleration changes.

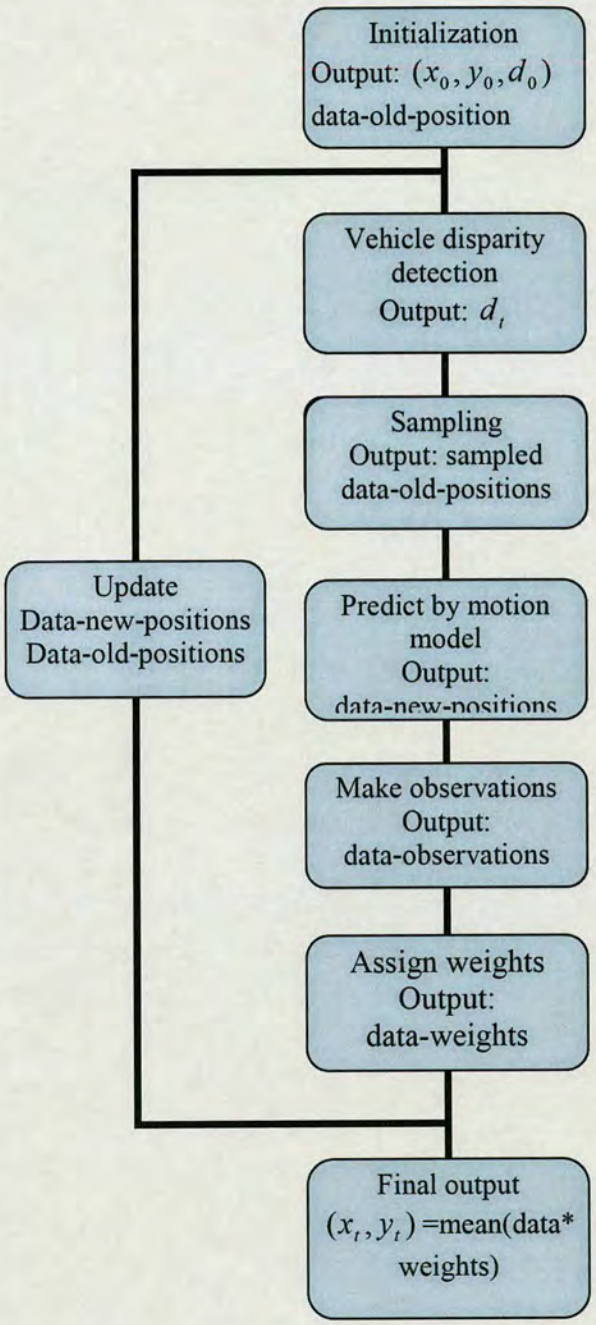
4.4.4 Condensation Algorithm

4.4.4.1 Initialization

As the motion model takes a second-order form, two frames should be initialized. For the first two frames, the same vehicle detection algorithm from the U-V-disparity vehicle detection algorithm is used. The results from the U-V-disparity vehicle detection algorithm are: position of the bounding box (x, y), height h, width w and disparity d. Then the colour histogram is calculated for the detected position. It is also ready for comparison with the particles in the first time step. Old particles (N particles) for this position are also generated by dispersing them according to a Gaussian distribution. Their weights are assigned with uniform distribution (equally weighted).

4.4.4.2 Running Condensation Filter

Flow chart



Algorithm

1. initialization
   initialize the position and disparity for the vehicle, calculate the scales for bounding box height and width, calculate colour histogram, generate particle with vehicle position according to Gaussian distribution, generate uniform weights for the particles
2. for each time step
   1). Detect the vehicle disparity in the current frame
   2). Sample the particles according to their weights
   3). Generate new positions by propagating old positions by the motion model
   4). Observe the colour histograms for the new positions, calculate the Bhattacharyya coefficient for each histogram with the histogram from previous step
   5). Assign weights for particles according their Bhattacharyya coefficients
   6). Estimate the mean state for the new positions according to their weights as output
   7). Update the positions with new positions, calculate the colour histogram for the estimated position for next time step

Table 4.3: Condensation algorithm

For each time step:

a) Detect the Vehicle Disparity

The detection of the vehicle disparity is very similar to the one in the initialization stage. The difference is that the one-dimensional scan range is smaller, as shown in **Figure 4.4**.

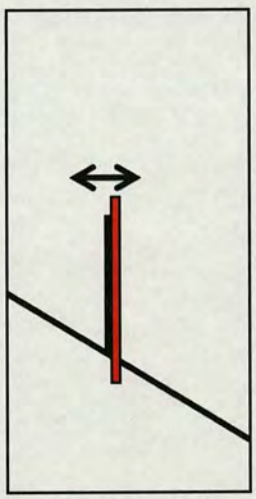


Figure 4.4: one-dimensional scan

The scan range is $[d_{t-1}-1, d_{t-1}+1]$ and the position and length of scan line are limited (the red line in **Figure 4.4**). The length of the scan line is decided by the size of the vehicle in the previous time step. It is relatively longer than the vehicle size from the previous time step result. Since the disparity change in two consecutive frames is assumed not to be larger than a pixel and the vehicle position change is no larger than a certain limit (20 pixels is used in the algorithm). So the position and length of the scan line should just cover the detected vehicle line with a 20 pixel variance. The disparity with the largest number of non-zero points is the disparity for the current frame.

b) Sampling

As the old positions and their weights are available, N particles $p_{t-1}^{(n)}$ and weights $\pi_{t-1}^{(n)}$, Calculate the normalized cumulative probabilities c'_{t-1}

$$cu_{t-1}^{(0)} = 0$$
$$cu_{t-1}^{(n)} = cu_{t-1}^{(n-1)} + \pi_{t-1}^{(n)} \qquad \textbf{(Equation 4.5 ~ 4.7)}$$
$$cu'^{(n)}_{t-1} = \frac{cu_{t-1}^{(n)}}{cu_{t-1}^{(N)}}$$

Generate a uniformly distributed random number $r \in [0,1]$

Find, by binary search, the smallest j for which $c'^{(j)}_{t-1} \geq r$

$$\text{Set } p'^{(n)}_{t-1} = p_{t-1}^{(j)} \qquad \textbf{(Equation 4.8)}$$

c) Generate New Positions

propagate each particle from the set p'_{t-1} by the motion model:

$$p_t = p'_{t-1} + A \cdot ([x_{t-1}, y_{t-1}]' - [x_{t-2}, y_{t-2}]') + R_t \qquad \textbf{(Equation 4.9)}$$

Where R_t is a multivariate Gaussian random variable

d) Make Observations

calculate the colour histogram for each particle of the set p_t

calculate the Bhattacharyya coefficient ρ for the colour histogram of each particle

$$\rho[h_{s_t}(n), h_{previous}] = \sum_{u=1}^{m} \sqrt{h_{s_t^{(n)}}^{(u)} \cdot h_{previous}^{(u)}} \qquad \textbf{(Equation 4.10)}$$

e) Assign the Weights

the weight for each particle of the set p_t

$$\pi_t^{(n)} = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{(1-\rho)}{2\sigma^2}} \qquad \textbf{(Equation 4.11)}$$

f) Estimate the Mean State of the Set p_t

$$E[p_t] = \sum_{n=1}^{N} \pi_t^{(n)} \cdot p_t^{(n)} \qquad \textbf{(Equation 4.12)}$$

4.5 Final Approach (U-V-Disparity and Modified Condensation)

This final approach is to use a single condensation filter to track multiple vehicles in the above sequence. This condensation algorithm is different from existing algorithm steps in that the algorithms are changed and particles are generated in a different way. It is able to fuse the disparity detection results, motion prediction and colour information together to give good multi-vehicle tracking results. Vehicles are detected in the thresholded U-V-disparity maps. Particles are generated with both the detection results and motion model prediction and their weights are assigned by a colour observation model.

Before adopting this final approach, several other techniques were tried. Those methods did not work adequately on this sequence. However, the final method consists of the good points of the different methods and some new ideas.

The aim of object tracking is to detect and locate the same object in a set of consecutive frames. Both of the approaches described above are able to fulfil this task. However, their performance is inadequate. If the still frame U-V-disparity vehicle detection algorithm is applied on each frame of the sequence, it can be considered as a tracking of the detected vehicles. In this way, it can be compared with the condensation tracking algorithm.

The condensation algorithm uses colour distribution to track vehicles when they are initialized. This tracking can be quite stable if the motion model is good enough. Since the motion model itself decides the new positions of the particles, the real position of the vehicle has to be close to the predicted new positions to allow a good tracking. However, as stated above, it is very difficult to get a single accurate motion model for different vehicles or even for one vehicle with different motions. At least, the motion model is not sophisticated enough without a training set. Another problem is that the algorithm is not able to recover from the errors which occur. As the particles are used repeatedly, when the tracking is lost, it is lost forever.

The U-V-disparity detection uses the disparity information. The disparity is 3D information, so the algorithm finds the disparity, position and the size of the vehicle at each frame. Good detection results depend on good thresholding of the U-V-disparity maps. Sometimes the position of bounding box is affected due to the noise in disparity map. A careful choice of threshold can reduce the noise and make the detected position not too far from the real position. But the best threshold for each frame is different and might require manual choice. However, since the detection is independent in each frame so that if the detection is totally wrong in one frame, it does not affect the next frame.

U-V-disparity method	**Condensation algorithm**
Uses disparity information	Uses colour information
Needs different thresholds for different frames	Needs different types of precise motion models
Noise affects the detection results	Noise does not affect much
Errors independent from each frame	Errors brought to the next frame
Slower	Faster

Table 4.4: comparison between U-V-disparity and Condensation algorithm approach

Both the U-V-disparity method and condensation algorithms can give quite good results if the conditions given in **Table 4.4** are satisfied. This is quite a high standard. In this final approach, the two methods are combined together to lower the individual standards, making the algorithm more automatic. The approach should be able to bound the vehicle more accurately, especially when the vehicle is not going straight

ahead, the size of the bounding box should change. And it should be able to track multiple objects, and handle the situation of emerging and disappearing objects.

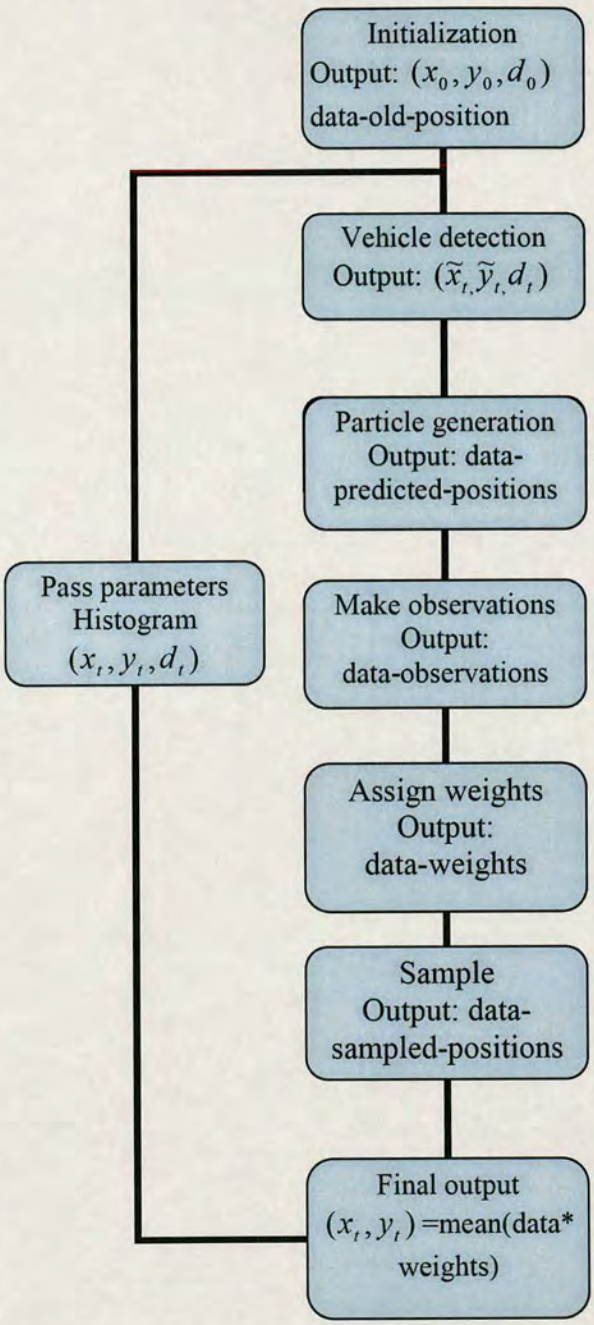
4.5.1 Initialization

The initialization step here is almost the same as the initialization in the normal condensation algorithm. The only difference is that three consecutive frames are initialized. For the first two frames, only the results of bounding box positions are calculated for motion predicted in the first time step. In the third frame, all parameters including bounding box position, width, height, disparity, and colour histogram are calculated. And the particular bounding box scales with respect to disparity are calculated as follow:

$$s_{width} = width / disparity$$
$$s_{height} = height / disparity \qquad \textbf{(Equation 4.13 ~ 4.14)}$$

Vehicles are divided into the two classes; going straight-ahead or not, with respect to the camera vehicle. A vehicle going straight-ahead shows only its rear in the image while a vehicle not going straight-ahead shows both rear and side. This is important because in a common multi-lane road scene, vehicles going straight-ahead are those that a driver is following in the same lane. Vehicles in other lanes such as overtaking, or turning vehicles are not going straight-ahead. What's more, the area of the vehicle side which is changing indicates the vehicle speed or change of direction. When a vehicle not going straight-ahead is cutting into the driver's lane or getting closer the visible area of the side will increase, which could be a warning to the driver of possible collision. Thus, it is reasonable to divide the vehicles into these two classes. The vehicles not going straight-ahead are more difficult to detect and localize in a proper size bounding box. This is because their sizes are changing and their motion is difficult to precisely model and they usually possess more than one value of disparity. This new combined tracking algorithm is dedicated to solving this problem.

Flow chart



Algorithm

1. initialization
   initialize the position and disparity for the vehicle,
2. calculate the scales for bounding box height and width, calculate colour histogram, generate particle with vehicle position according Gaussian distribution, generate uniform weights for the particles
3. for each time step
   vehicle detection: detect the vehicles in U-V-disparity map, find their positions and decide their status by motion model
   particle generation: generate N particles, half them using detection results and the other using motion prediction
   1).observation: observe the colour histograms for the particles, calculate the Bhattacharyya coefficient for each histogram with the
   2).histogram from previous step
   3).assign weight: assign weights for particles according their Bhattacharyya coefficients
   4)sample: sample the particles according to their weights
   5)output: estimate the mean state for the new positions according to their weights as output
   6)pass the results and colour histogram to the next time step

Table 4.5: modified Condensation Algorithm

4.5.2 For Each Time Step

a) The Vehicle Detection Algorithm

The vehicle detection is the most important part in this whole vehicle tracking algorithm. It is greatly different from the vehicle detection algorithm described above. It includes more sophisticated vehicle disparity detection, vehicle status decision and vehicle position prediction.

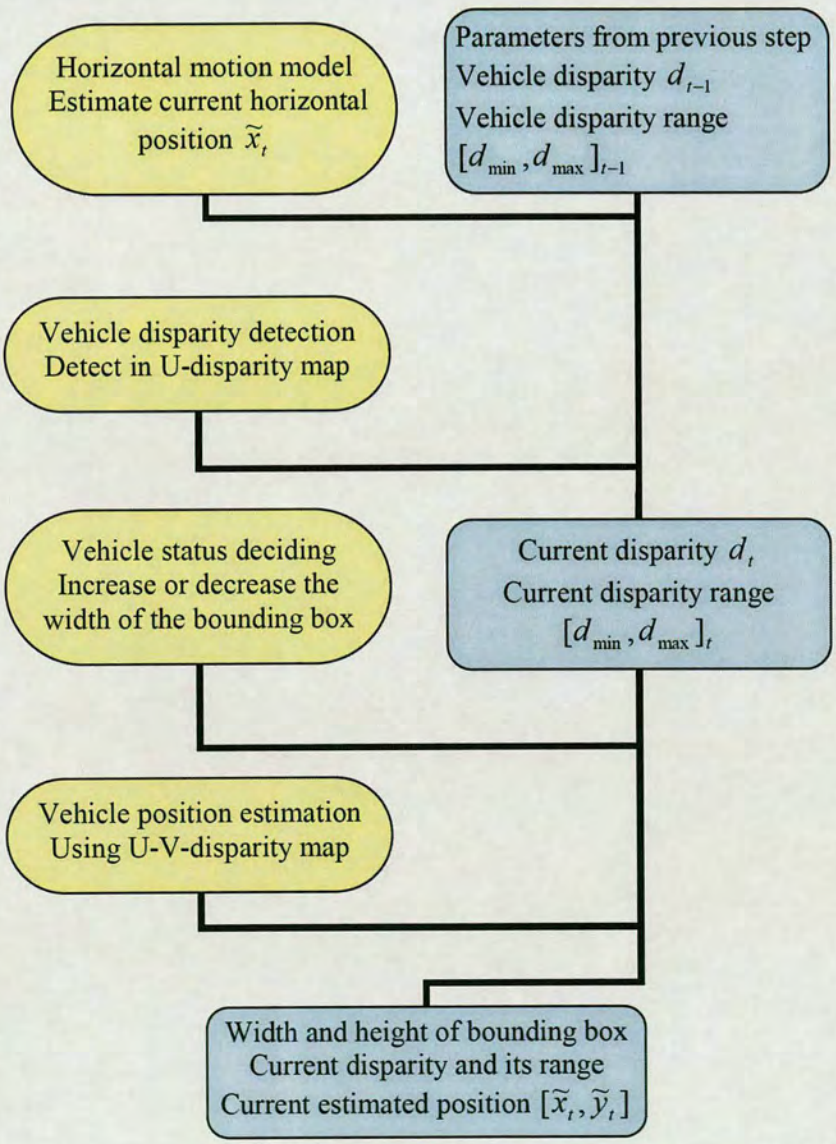


Figure 4.5: vehicle detection algorithm

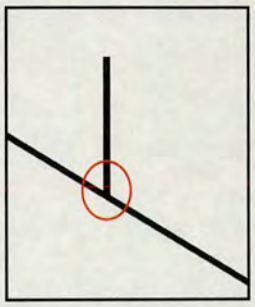


Figure 4.6: the contact point of vehicle and road surface

a1) horizontal motion model

From experience of the normal condensation tracking, it was clear that the vertical position of the bounding box was easier to find than the horizontal position. The algorithm finds the lower end of the bounding box which is simply the contact point of the vehicle with the road in the V-disparity map (**Figure 4.6**). When the disparity of the vehicle is found, the contact point is quite easy to find and is more stable with respect to noise and the position does not change for different thresholds of the V-disparity. A single threshold for the whole sequence can be used.

However, the horizontal position is vulnerable to noise. The bounding box is required to surround the whole vehicle inside it. This is very difficult when the vehicle is not going straight-ahead, especially when it is turning, as its horizontal size increases dramatically.

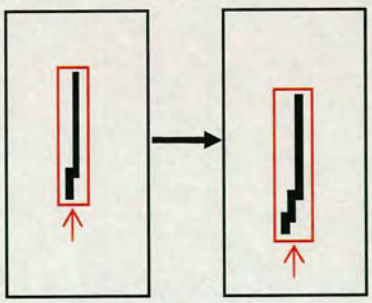


Figure 4.7: bounding of a vehicle not going straight-ahead in U-disparity

Figure 4.7 shows what a not going straight-ahead vehicle looks like in a U-disparity map. The upper end position change in the U-disparity map is due to the vehicle's horizontal movement while the lower end change is due to vehicle movement and size change as well. For a proper bounding box for the vehicle, the lower end (where the arrow points in **Figure 4.7**) needs to be found. This end is vulnerable to noise since it is the turning end of the vehicle, it is further from the camera and fewer points are showing in the U-disparity map. This means the non-zero pixels in the region of interest (inside the red rectangular) should quite closely represent the true vehicle occupation. Simply increasing the width of the bounding box from the previous time step to define the region of interest is inaccurate, so a horizontal motion model which models the motion of the turning end of the vehicle is used to reduce this region. The motion model takes horizontal positions from three previous time step to predict the current position in the current step.

Assuming:

$$x_{t-2} - x_{t-3} = x_{t-1} - x_{t-2} + \beta \qquad \textbf{(Equation 4.15)}$$

$$x_{t-1} - x_{t-2} = \tilde{x}_t - x_{t-1} + \beta \qquad \textbf{(Equation 4.16)}$$

Thus,

$$\tilde{x}_t = 2 \cdot x_{t-1} - x_{t-2} - \beta \qquad \textbf{(Equation 4.17)}$$

$$\beta = 2 \cdot x_{t-2} - x_{t-1} - x_{t-3} \qquad \textbf{(Equation 4.18)}$$

$\tilde{x}_t$ is the predicted turning end of the vehicle.

a2) Vehicle Disparity Detection

The vehicle disparity detection is designed to detect vehicles with different status, going straight-ahead or not going straight-ahead. A not going straight-ahead usually has more than one disparity value, as seen in **Figure 4.8**:

Figure 4.8: not going straight-ahead in U-disparity

To detect the whole disparity occupation of a vehicle not going straight-ahead, the vehicle disparity detection algorithm allows a vehicle to have a disparity range instead of a single value of disparity and the disparity of the vehicle is calculated to sub-pixel value. For example:

$$d_{range} = [d_{\min}, d_{\max}],\ d_{vehicle} = \frac{1}{n_d} \cdot \sum_d d \cdot n_d , \qquad \textbf{(Equation 4.19)}$$

in which n_d is the number of non-zero points for the disparity d. Since the resolution of the image is low and the baseline of the stereo cameras is small, a single disparity value covers a certain range of distance (about 1.5~2 meters) so that there may be a step change in the size of the bounding box, which is decided by the disparity, if the disparity changes. The sub-pixel disparity calculation helps reduce the effect of step change. It helps when the vehicle is not going straight-ahead (since no side part of the vehicle can be seen for a vehicle going straight-ahead, the range of the disparity is always single value).

At each time step, the vehicle disparity detection algorithm takes the results (disparity d_{t-1}, disparity range $[d_{\min}, d_{\max}]_{t-1}$ and the width of the bounding box w_{t-1}) from the previous time step. In the current U-disparity map, the scan range becomes $[d_{\min} - 1, d_{\max} + 1]$. For the predicted turning end of a vehicle $\tilde{x}_t$, the region of interest becomes $[\tilde{x}_t - w_{t-1} - a, \tilde{x}_t + a]$, where a is a small amount of the expansion in the predicted vehicle region (set as 5 pixels in the algorithm). The numbers of the

non-zero points for each column in the region are saved in $[n_1,...n_{w_{t-1}+2a}]$. The disparity range for the current time step is defined as the continuous interval of $n_i > b$, (b is set as 10 pixels in the algorithm). With the current detected disparity range and the numbers of non-zero points, the current vehicle disparity can be calculated according to **Equation 4.18**.

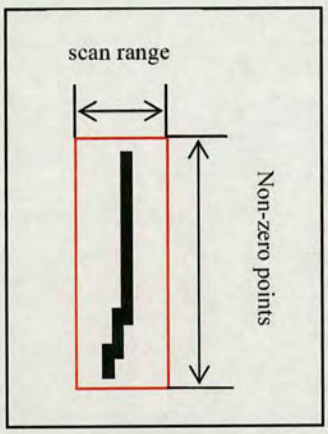


Figure 4.9: U-disparity scan range

This vehicle disparity detection also handles vehicles emerging and disappearing. The handling of the disappearing vehicles is easier. When all the saved numbers of non-zero points of each column in the interested vehicle region are smaller than b, the vehicle has disappeared. On the other hand, the handling of emerging vehicle needs a vehicle redetection in the U-disparity map since a vehicle is always visible in the U-disparity map if it is not totally occluded.

As shown in **Figure 4.10**, the left vehicle in the disparity map is visible in the U-disparity map but not in the V-disparity. The vehicle redetection is performed after the vehicle disparity detection. All the points which belong to the existing vehicles in the U-disparity map are eliminated, the same vehicle detection algorithm in the initialization stage is performed here to find a possible emerging vehicle and calculate the parameters for it.

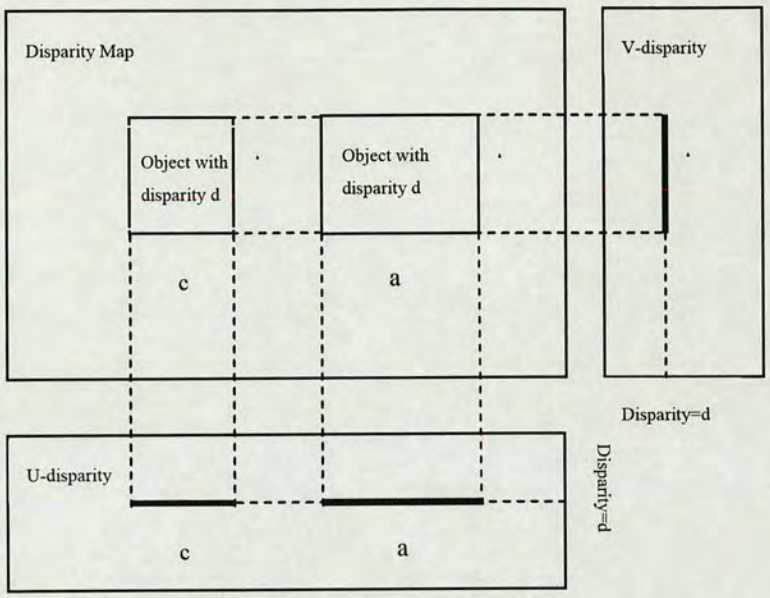


Figure 4.10: two vehicles with occlusion in V-disparity and U-disparity map

a3) Vehicle Status Decision and Localization

In this step, the detected vehicles are classified as going straight-ahead or not going straight-ahead. The status is decided by calculate the percentage of the total vehicle pixels which are side pixels. In a detected disparity range $[d_{\min}, d_{\max}]$, side pixels are those with disparity range $[d_{\min}, d_{\max} - 1]$. If the percentage of side pixels is larger than 10 percent, the vehicle is classified as not going straight-ahead. As stated above, in the vehicle localization process, the vertical position finding is easier. With the detected disparity, the vertical position is found as the contact point of the vehicle and the road in the v-disparity map. In finding the horizontal position, the algorithm takes the same range as in the vehicle disparity detection algorithm, with scan range $[d_{\min} - 1, d_{\max} + 1]$ and non-zero pixels counting region $[\widetilde{x}_t - w_{t-1} - \beta, \widetilde{x}_t + \beta]$. The point with the largest column number is the horizontal position of the vehicle. If the vehicle is classified as not going straight-ahead, the algorithm needs to decide the change in the width of the vehicle. The width in the current time step w_t is calculated

in this way: $w_t = w_{t-1} + \hat{x}_t - \tilde{x}_t$, $\tilde{x}_t$ is the predicted turning end of the vehicle, $\hat{x}_t$ is the detected turning end of the vehicle.

b) Particle Generation

In this modified condensation algorithm, the particles are generated in each time step. The particles for vertical and horizontal position are generated separately. For the vertical position, the detection results are quite stable, they are quite close to the real positions. Thus the particles for the vertical position are generated by dispersing the detected position with a Gaussian distribution. For the horizontal position, the detection results are not as good as the vertical position, especially when a vehicle is not going straight-ahead. The particles are generated from both detection results and also from previous results. Half of the particles are generated in the same way as the vertical ones. The other half are generated by dispersing the position predicted by a horizontal motion model which takes previous two positions into account. This can reduce the error from a poor horizontal detection. Using this method of particle generation, the particles are propagated not only by motion model but the detection results as well. The problem of requiring a precise motion model is greatly reduced. This makes the algorithm less susceptible to errors.

c) make observations

for the generated particles p_t',

calculate the colour histogram for each particle of the set p_t'

calculate the Bhattacharyya coefficient for the colour histogram of each particle, as **Equation 4.10**.

d) Assign Weights

the weight for each particle of the set p_t', as **Equation 4.11**.,

e) Sample

As the particles and their weights available, N particles $p_t^{(n)'}$ and weights $\pi_t^{(n)'}$,

Calculate the normalized cumulative probabilities c'_t, as **Equation 4.5 ~ 4.7**.

Generate a uniformly distributed random number $r \in [0,1]$

Find, by binary search, the smallest j for which $c_t'^{(j)} \geq r$

Set ,

$$p_t^{(n)} = p_t^{(j)'}, \pi_t^{(n)''} = \pi_t^{(j)'} \qquad \textbf{(Equation 4.20 ~ 4.21)}$$

Then

$$\pi_t^{(n)} = \frac{\pi_t^{(n)''}}{\sum_n \pi_t^{(n)''}} \qquad \textbf{(Equation 4.22)}$$

f) estimate the mean state of the set p_t, as Equation 4.12.

4.6 Conclusion

The final vehicle detection and tracking approach has been introduced in this chapter. It was also compared with basic U-V-disparity and Condensation approach. Given the complexity of the data sequence, such as multi-vehicle appearance, various vehicle motion and vehicle emerging and disappearing, the basic U-V-disparity and Condensation algorithm are not adequate for the vehicle detection and tracking. The final approach is dedicated to the complexity of the sequence which could happen in various road environments.

The vehicles in the sequence are divided into two classes: going straight ahead and not going straight ahead. The U-V-disparity detection in the approach is unique, it uses a one dimensional motion modal to limit the detection region, and able to detect these two classes of vehicles and find the corresponding disparity. Based on this, a modified Condensation algorithm is used to track the detected vehicles. The sampling process is modified to take both detection and previous time step results into account which is able to reduce the degeneration in the normal Condensation algorithm.

In the next chapter, results from these algorithms are carried out. Comparison is also made on the results.

5 Results and Discussion

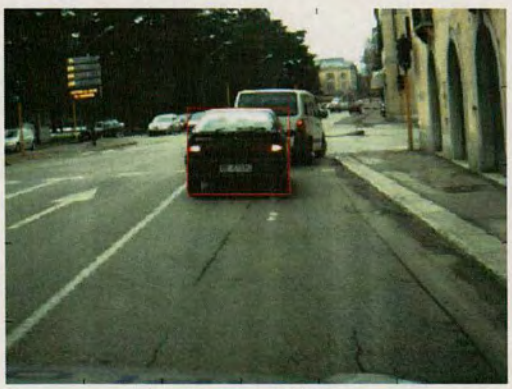
In this chapter, the results of using different methods on the sequence will be presented. Comparison will be made between the results of a normal condensation algorithm and the final approach adopted.

5.1 Normal Condensation Algorithm

The normal condensation algorithm was tested on the sequence. The condensation algorithm was used to track all three vehicles but separately. The initialization of the vehicle is done with the U-V-disparity vehicle detection presented in the previous chapter. The results are presented every 5 frames from a total of 50 frames.

5.1.1 Tracking the black car with second order motion model without hand tuning

Frame 5222



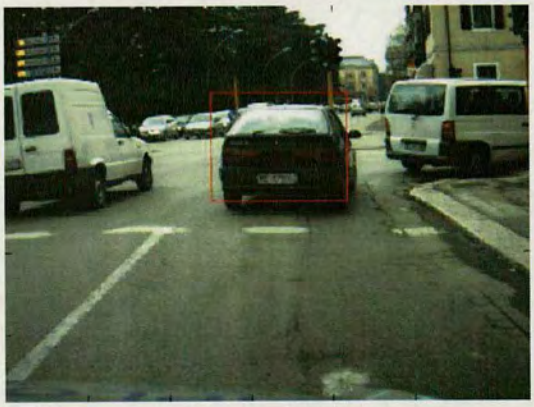
Frame 5227

Frame 5232

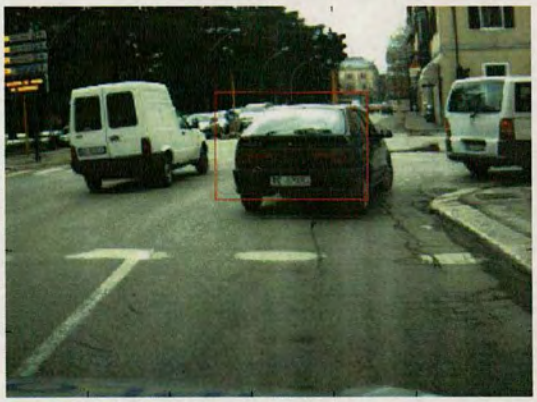
Frame 5237

Frame 5242

Frame 5247

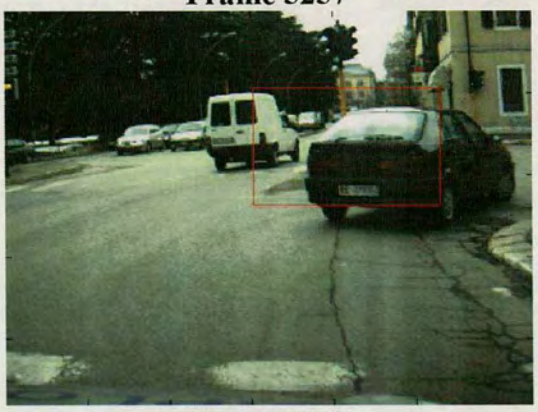


Frame 5252

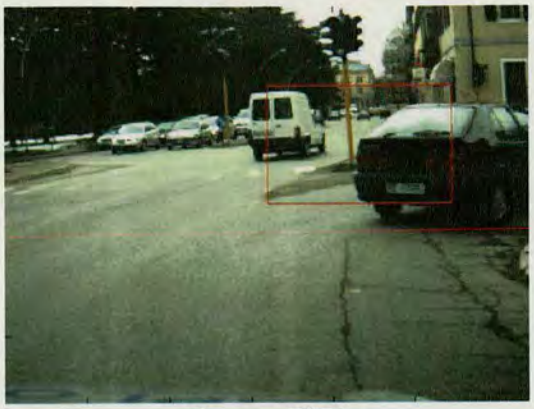


Frame 5257

Frame 5262



Frame 5267



Frame 5271

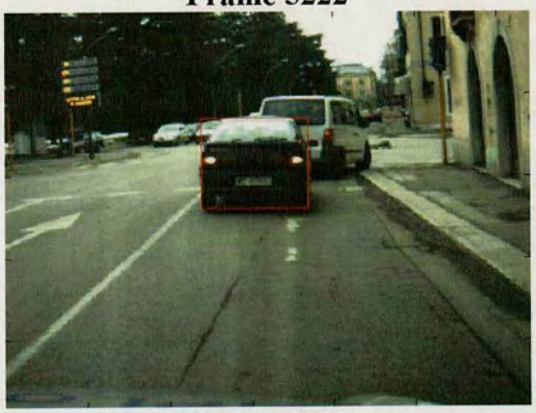
Figure 5.1: Tracking Results of tracking the black car using the Normal Condensation Algorithm and a second order motion model without hand tuning

The results presented in **Figure 5.1** show tracking of the black car. In the first 20 frames, this car is going straight ahead. It then starts to turn to the right. As mentioned in the algorithm description in the previous chapter, the quality of the tracking results depends greatly on the motion model. The motion model used here is a second order motion model. As can be seen in **Figure 5.1**, when the car is going straight ahead, the tracking is perfect. The main reason for this is that the vehicle is not moving much between each frame (only a few pixels to the right). Thus the motion model does not affect the predicted position a lot, since the change between two consecutive frames is small and the scale for the motion change is set to be small. The particles used are able to include the true position and sampling is able to pick up the particles close to the true position. When this car starts to turn more rapidly, the motion starts to change dramatically, especially in the horizontal direction. The small scale factor of the motion model is not adequately able to follow the true motion. The particles predicted by the motion model are therefore quite far from their true position. Even if the sampling picks up particles which are close to the true position, they are still too far from where they should be. This results in an inaccurate location in one frame. However, the design of the conventional condensation algorithm means that this inaccurate location is used in the next time step, making the error in the next location even worse. In this situation, the tracking of the vehicle is totally lost. As can be seen in the images in **Figure 5.1**, this happens shortly after the black car starts to turn sharply to the right.

5.1.2 Tracking the black car with hand-tuned motion model

Frame 5222

Frame 5227



Frame 5232

Frame 5237

Frame 5242

Frame 5247

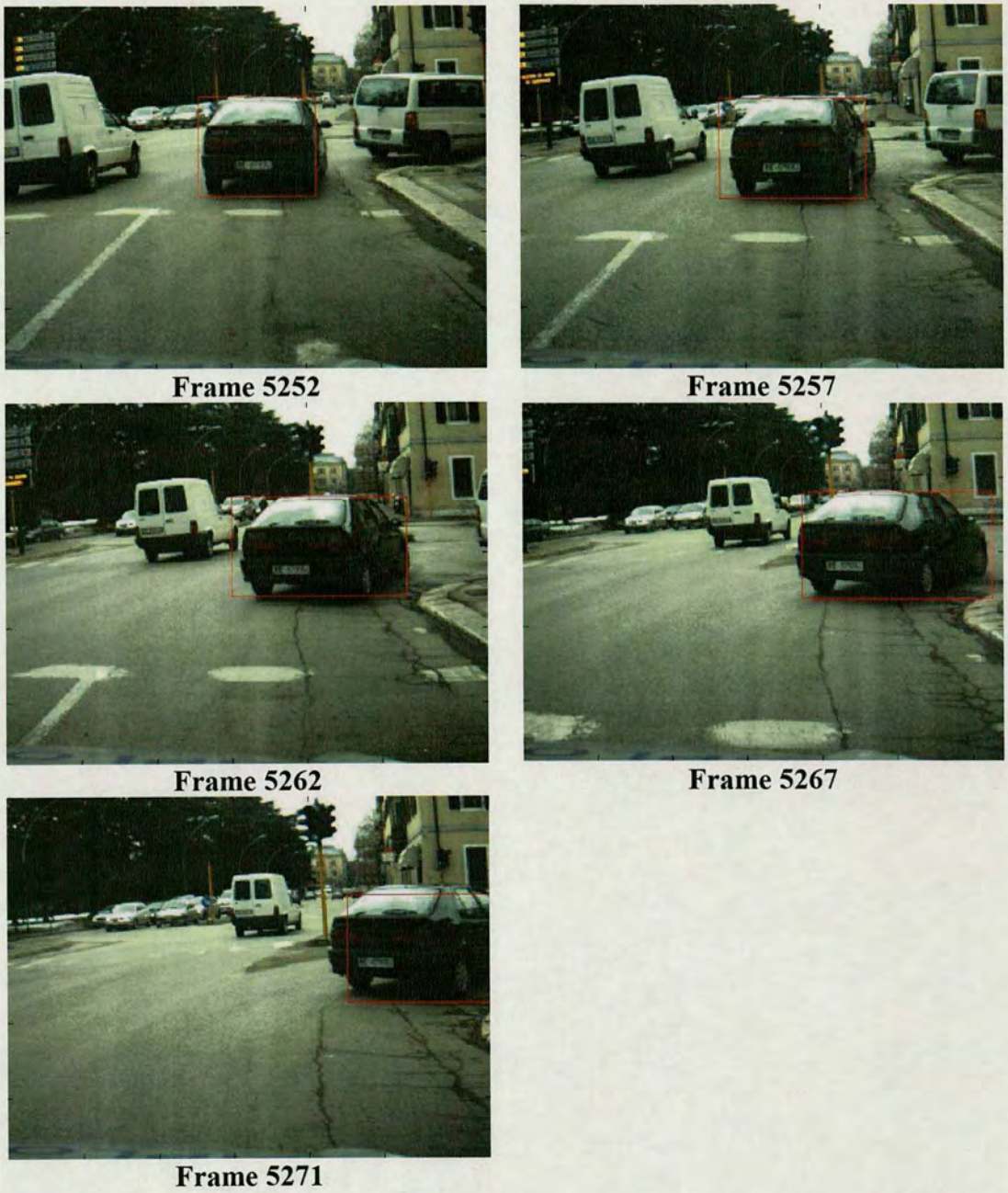


Figure 5.2: Tracking Results of tracking the black car using the Normal Condensation Algorithm and a hand-tuned motion model

The results presented in **Figure 5.2** are from tracking with same motion model, but in this case the scale factor of the motion model has been tuned by hand: for the first 20 frames, the scale is set to a small value, while in the later frames the scale factor is increased by a small amount in each frame, which models the speed increase of a second order acceleration. By doing this, the motion predicted by the motion model is closer to the real motion of the car, which improves the tracking of the turning part

of the sequence. However, this motion model is still not perfect. It lags the real motion by a small amount. When the car just starts to turn, the motion model does not "notice" that change. It finds out and catches up a few frames later. With hand-tuning of the scale factor of the motion model, the normal condensation algorithm works well when tracking one vehicle in the sequence. The next two sets of results are from the tracking of the turning van and the overtaking van.

5.1.3 Tracking the turning van with hand-tuned motion model

Frame 5222

Frame 5227

Frame 5232

Frame 5237

Frame 5242

Frame 5246

Figure 5.3: Tracking Results of tracking the turning van using the Normal Condensation Algorithm and a hand-tuned motion model

The results of the turning van tracking are presented in **Figure 5.3**. The normal condensation algorithm applies to 25 frames of the sequence, before the van is occluded in the image. With the hand-tuned motion model, the tracking of the van is quite accurate. Although the van is partially occluded by the car, the algorithm is still able to track it with the chosen motion model and its colour histogram.

5.1.4 Tracking the overtaking van with hand-tuned motion model

Frame 5246

Frame 5251

Frame 5256 Frame 5261

Frame 5266 Frame 5271

Figure 5.4: Tracking Results of tracking the overtaking van using the Normal Condensation Algorithm and a hand-tuned motion model

The results of the tracking of the overtaking van are presented in **Figure 5.4**. The overtaking van is moving at quite a speed, so that the scale factor of the motion model is much bigger than that for the turning van and car.

The results presented in this section show that the normal condensation algorithm is able to track single vehicle. However, the quality of the tracking is greatly affected by the accuracy of the motion model. It is difficult to find a precise motion model without a training set. In this algorithm, the parameters of the motion model require to be tuned by hand, which restricts the algorithm to a particular object in the sequence. The size of the bounding box also affects the quality of the tracking, since the colour histogram is calculated for what is inside the bounding box, but it does not affect the results as much as the motion model. Here, the size of the bounding box is decided by the scale and the disparity. The scale is calculated during the initialization.

If the pose of the vehicle is constant, the size of the vehicle is purely decided by its disparity. However, if the pose of the vehicle changes from when it is initialized, the scale size should change. In this vehicle tracking application, the vertical size scale does not change much, while the horizontal size scale changes when the motion of the vehicle changes. The horizontal size scale was also tuned by hand for these results.

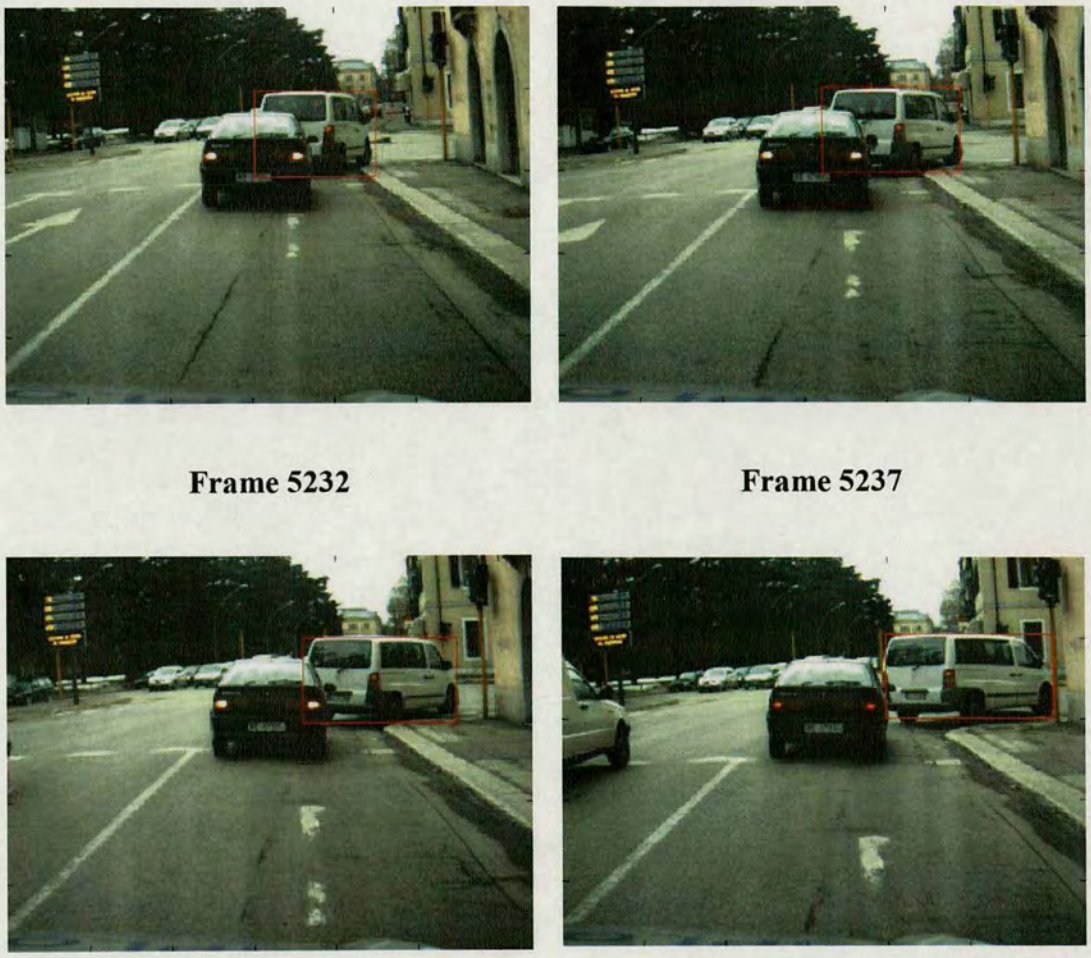
5.2 The Modified Condensation Algorithm

As mentioned above, the motion model chosen is important for successful condensation tracking and selecting a suitable one is difficult. It was therefore decided to discard the motion totally and instead use the U-V-disparity detection results as the prediction. As explained in **Chapter 3**, the condensation algorithm was also changed to suit this change in prediction method. This modified algorithm was first tested on a single vehicle. The turning van was chosen, because it is the most difficult one to track, since it has both occlusion and rapid motion change.

5.2.1 Tracking the black car using U-V-disparity detection as prediction

Frame 5222

Frame 5227



Frame 5232 **Frame 5237**

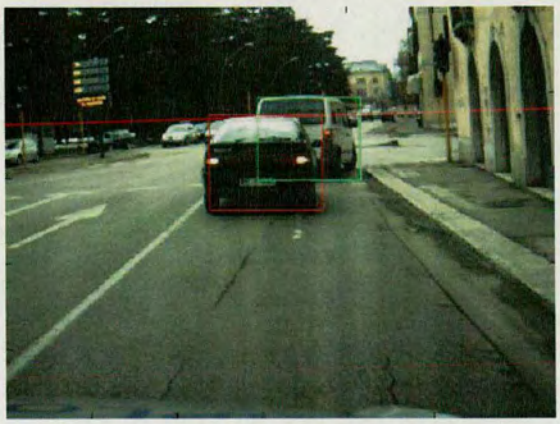
Frame 5242 **Frame 5246**

Figure 5.5: Tracking Results for tracking the turning van using the Modified Condensation Algorithm

The results of applying the modified condensation algorithm are presented in **Figure 5.5**. As can be seen, these results are very good. The van is tracked well and the bounding box fits properly. These results at least as good as those for the normal condensation algorithm. However, the huge advantage here is that no motion model is needed and no parameters need to be tuned by hand. This means the same algorithm can track any of the three vehicles in the sequence without tuning the parameters and despite their significantly different motion characteristics.

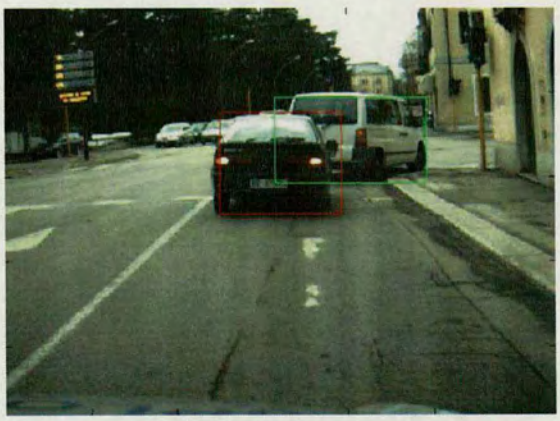
5.2.2 Tracking all 3 cars using U-V-disparity detection as prediction

Frame 5222

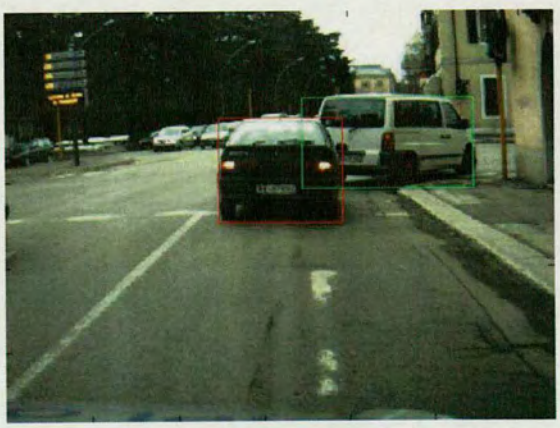


Frame 5227

Frame 5232

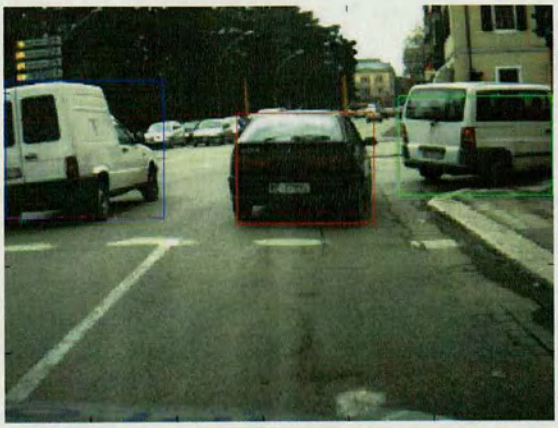


Frame 5237



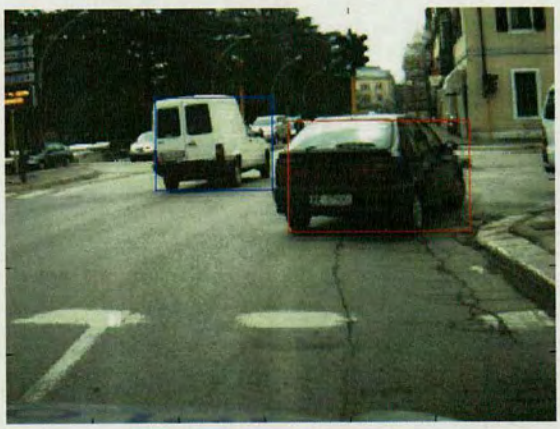
Frame 5242

Frame 5247



Frame 5252

Frame 5257



Frame 5262

Frame 5267

Frame 5271

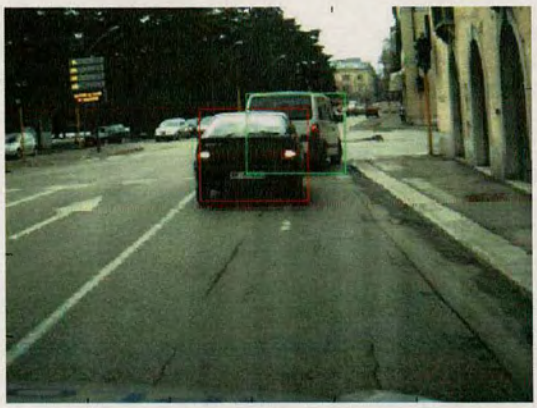
Figure 5.6: Tracking Results for tracking 3 vehicles using the Modified Condensation Algorithm

The results of the tracking 3 vehicle at same time are presented in **Figure 5.6**. These are the results tracking 3 vehicles using the modified condensation algorithm with no prior motion model. As noted above, the results of U-V-disparity detection are used as the position prediction. As can be seen from these images, the tracking of the turning vehicle is actually better than that of the other 2 vehicles, especially in finding the turning end of the vehicle. There is a reason for this. In the U-V-disparity detection, the disparity pixels are projected horizontally and vertically. Thus the detection can only find the horizontal and the vertical boundaries of the vehicle. By finding these boundaries, the model of the vehicles is already selected to be a rectangular bounding box. In this sequence, the shape of the turning van in the image is more like a rectangle than the other two. Thus the model best fits the turning van. For the other two vehicles, if they are going straight ahead, their shapes are almost rectangular. But when they start to turn, the more tapered shape of the turning end makes the overall shape less rectangular. This more tapered end also makes the detection in the U-disparity map rather unstable since fewer pixels are projected horizontally. This problem happens in frame 5267. What's more, when it happens, the colour histogram measure is not strong enough to "pull" the bounding box back to the right position since the tapered end also makes the colour histogram measure more vulnerable (there is more area in the bounding box which does not belong to the vehicle). However, this only happens when the vehicle is actually turning and with this modified condensation algorithm, it only affect the current frame not all the successive frames (as happens to the normal condensation algorithm).

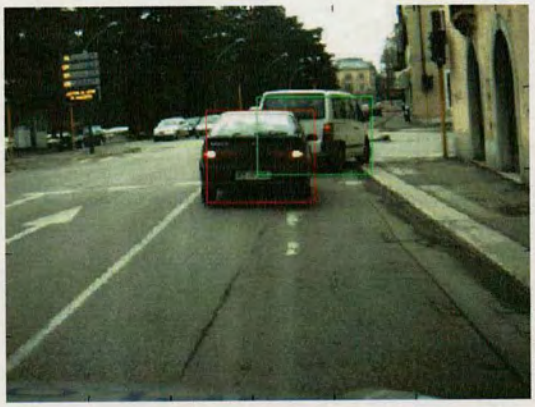
However, this problem can be solved or at least the effect minimised if the prediction is better. So in the final approach the horizontal position prediction is made by both U-disparity detection and a horizontal motion model (**Equations 3.15 ~ 3.16**).

5.2.3 Tracking all 3 cars using motion model and U-V-disparity detection combined prediction

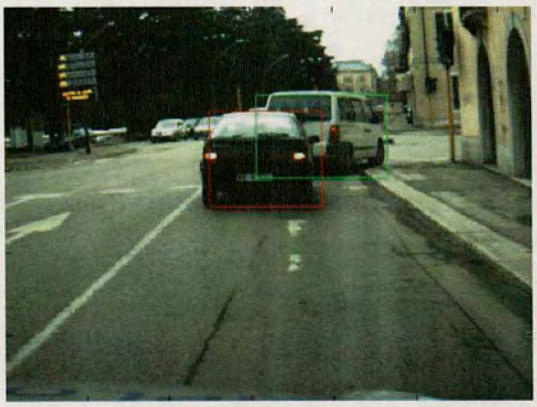
Frame 5222



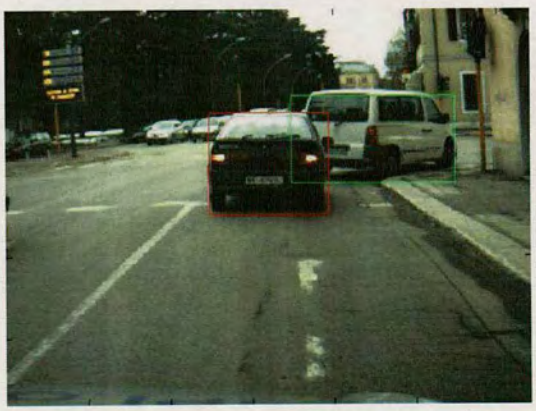
Frame 5227



Frame 5232

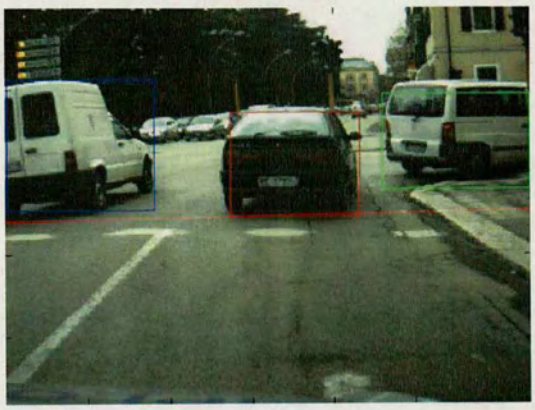


Frame 5237

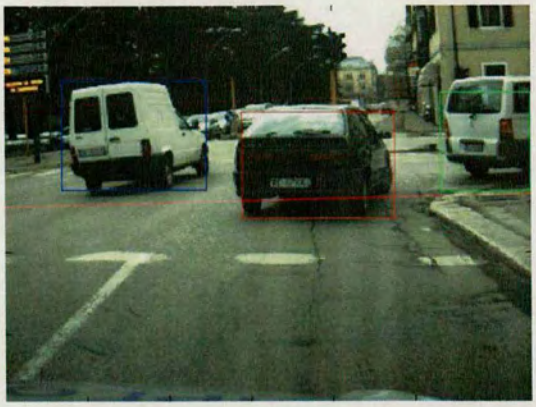


Frame 5242

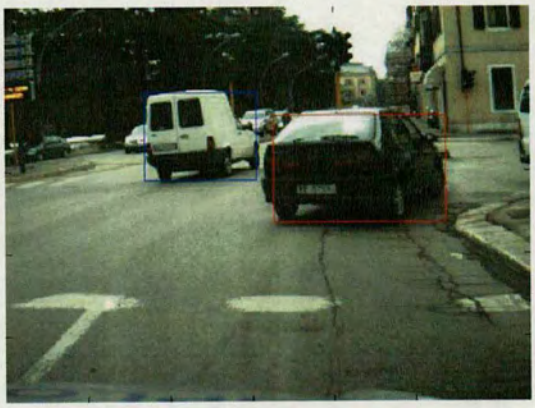
Frame 5247



Frame 5252



Frame 5257



Frame 5262

Frame 5267

Frame 5271

Figure 5.7: Tracking Results for tracking 3 vehicles using the final Modified Condensation Algorithm

The results for this final approach are presented in **Figure 5.7**. As can be seen from these images, with combined U-disparity detection and horizontal motion model prediction, the tracking is even better when the vehicles are turning (see **Frame 5267**). As mentioned in **Chapter 3**, the combined prediction is realized by generating half of the particles using U-disparity detection and the other half by using the motion model. This makes the algorithm more stable and avoids losing tracking completely.

5.3 Conclusion

Comparison has been made between tracking results using normal and modified Condensation algorithm. The first set of results show that the normal Condensation algorithm is able to track under the condition of using precise motion model. The prediction in the algorithm is based on the motion model. However, for this particular data sequence, it is very difficult to find a motion model that does not need hand-tuning. The inaccurate motion model gives inaccurate prediction which results in bad tracking performance, sometimes even losing the tracking.

Then the results of tracking using the modified Condensation algorithm with U-V-disparity prediction are presented. With the accurate U-V-disparity detection, the tracking is at least as good as when using a hand-tuned motion model. However, if any detection problems happen, the tracking is affected in that frame. This sometimes makes the total tracking process not smooth.

Therefore, the final solution is to modify the normal Condensation algorithm in the way that particles are generated (prediction) by both motion modal and U-V-disparity detection. The motion modal used here is same as the one used in the U-V-disparity detection. The parameters of the motion model are calculated at each time step based the positions in the previous two frames. Particles are generated (prediction) based on both motion model prediction and U-V-disparity detection. Thus, the inaccuracy of the motion model is greatly compensated by the U-V-

disparity detection, which allows more particles to cover the real position of the vehicle. In the other hand, the smoothness of the tracking remains by the motion model. The results of tracking all 3 vehicles using this modified Condensation algorithm show good tracking performance, all 3 vehicles are tracked, well bounded in the boxes, and vehicle emerging and disappearing are also handled.

The overall summary and conclusion of this method well be presented in the next chapter together with some future work suggestion regarding this project.

6 Conclusion and future work

6.1 Conclusion

This thesis describes a method of on road vehicle tracking. A stereo sequence is used in this method. Vehicles are detected by U-V-disparity detection, which is also used as the prediction of a modified Condensation algorithm. The results are satisfactory, various types of vehicles are detected and tracked, different situation including vehicle turning, emerging and disappearing are handled.

6.1.1 Nature of the Data

The data sequence used in this project is a DIPLODOC data sequence [48] which was taken in Italy including some complex urban road environments. This data sequence is a good choice to test a vehicle detection and tracking algorithm since it contains some typical complex on road situations. There are 3 vehicles which appear in the whole sequence in total. One of them is a black car in the nearest position which turns into the right and goes off late in the sequence. Then in front of the black car, a white van which is partially occluded by the black car also turns to the right and goes off. There is another overtaking van with a high speed emerging in the middle of the sequence. This is quite a typical complex on-road situation with various vehicles and motions.

This data sequence was taken by a pair of stereo cameras mounted on a moving vehicle. The resolution of the stereo images is quite low (320×240). This is very common for a stereo camera system used in an on road vehicle tracking system in order to minimize the cost of the system on an industrial implementation basis.

6.1.2 Data Processing Problems

The presented vehicle detection and tracking method handles some difficult situations. Due to the low resolution of the data sequence, difficulty rises in the vehicle detection through a disparity map. Because of the data sequence, the calculated disparity map is inaccurate and also is low resolution. As discussed in the previous chapters, the resulting disparity map is sparse. All these make the direct vehicle detection through the disparity map very challenging. The U-V-disparity detection used in this project is aimed at solving these problems. It provides a unique way of using the sparse disparity map by making the search of vehicles one dimensional. Vehicles are successfully detected and bounded into a rectangular bounding box.

This method also handles vehicle overlapping through the U-V-disparity detection. The turning van is partially overlapped by the black car for almost half of the sequence. Since the vehicle is only partially overlapped, it is still visible in the disparity map. However, the visible part of the vehicle in the disparity map is not significant enough to be detected directly. The U-V-disparity method is able to concentrate these disparity pixels onto vertical and horizontal directions, which makes them significant enough for detection. The partial overlap then can be located if only either end of the vehicle is found in the U-disparity and V-disparity. The vehicle can be bounded by using the size of the bounding box in the previous time step. For the situation of totally overlapping, in other words, total occlusion, the vehicle is treated as a disappearing vehicle. When it becomes visible again, it is treated as new emerging vehicle.

Vehicles emerging and disappearing is dealt with during the vehicle detection. When the length of the visible part in the U-V-disparity is under 20 pixels, the vehicle is treated as disappearing. The choice of a 20 pixels threshold is adequate since the search range is limited by the vehicle position in the previous time step, which

greatly lower the chance of interfering with noises in the U-V-disparity. The emerging vehicle is detected by a quick overall search in the U-disparity, at each time step which is similar to the search in the initialization. It is able to detect new emerging vehicles, but increases the computational load.

Turning vehicle detection and tracking are always challenging. This method is adequate to handle this due to the simplicity of the object model and classification of the vehicles. Turning vehicles appear differently from their going-ahead appearance. This sometimes causes loss of tracking since the turning appearance is treated as a new object. In this method, the object model used is a very simple rectangular bounding box. The objective here is to detect the vehicle and fit it totally into the bounding box. The algorithm classifies the vehicle into going-ahead and not going-ahead through the U-V-disparity detection, which allows better prediction of the vehicle boundaries, thus better bounding box size and position.

All of the above mentioned situations are handled through the U-V-disparity approach. The normal Condensation algorithm is thus modified to suit this U-V-disparity approach. The sampling process is modified to sample from both previous time step and U-V-disparity results. This achieves better tracking performance and continuity.

6.1.3 Performance

This method is implemented in Matlab. The total algorithm, including disparity map calculation and Condensation algorithm, can not run in real time running in Matlab. However, the disparity map calculation method used here is 5×5 block matching algorithm, which is claimed to be implemented in hardware [42, 43]. The Condensation Algorithm can also be a real time algorithm which can be implemented in C++. In this project, no image processing tool box is used in Matlab, this makes it

easier to implement the algorithm in C or C++. So based on these facts, this whole algorithm could probably run on real time with a hardware and C++ implementation.

5.1.4 Possible Application

Clearly this project is aiming at a driver-assistant/driving-automation system. On-road vehicle detection and tracking is just the first step. The total driver-assistant/driving-automation system should also include pedestrian detection, lane detection and following algorithm and directional guidance etc. The high reliability level of the system is required. For the moment, the most possible application of this approach is the adaptive cruise system. This is usually used in a high way scenario. The normal cruise system is to set the vehicle to run a pre-set speed, while the adaptive cruise system can warn the drive or lower the speed when there are other vehicles in near front. This requires the calculation of the relative speed and distance between vehicles, which is part of the future work of this project, and will be introduced in the following section.

6.2 Future Work

6.2.1 For the Near Future

The first step for the future work of this project is to implement it in real time. As discussed above, there is a good chance that this algorithm could run in real time. The basic idea is to implement disparity calculation in hardware and Condensation algorithm in C++. However, one major fact which could effect the real time running performance is that the U-V-disparity calculation is based on histogram calculation,

whose calculation speed is greatly dependent on the size of the stereo images. This is a possible constraint of this project.

To utilise the outputs of the project, some detailed calculation is required, including vehicle relative speed, position etc. These calculations can be achieved based on tracking results from each time step. With these, the track of the tracked vehicle can be calculated, which could be used to better predict the vehicle motion in the next time step. If this project is part of a driver-assistant/driving-automation system, some decision making algorithm should also be developed based on these calculated results.

6.2.1 For the Far Future

In this project, all vehicle detection and tracking algorithms are based on stereo cameras. As discussed in the previous sections, the stereo cameras used are low cost, which is very important from an industrial perspective. However, low cost cameras provide low quality images, especially low resolution. Although the low resolution problem can be partially solved by using a simple object model and U-V-disparity detection, a more sophisticated object model such as active contour, demands better quality images. High resolution images greatly increase the cost of cameras and increase computational load. One possible solution to this such as [58], is to use a stereo and other range instruments combined method, such as Lidar [56,57] and Range camera [01,02,03]. Lidar and range camera are active vision systems for revealing 3D information. Compared to the stereo system, they provide higher resolution but with a smaller view angle. In the future, a system which uses stereo and Lidar/range camera combined techniques could be a layered system. High level detection could be achieved by stereo techniques while for the regions of interest, detailed information could be revealed by Lidar/range camera techniques. This combination takes the advantage of the stereo camera's wide view angle and also lowers the computational load caused by increased resolution by identifying

particular regions of interest.

References

[01] T. Oggier et al., "An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRangerTM)", SPIE Proc Vol. 5249-65, St. Etienne (2003)

[02] Fritzsche M., Prestele C., Becker G., Castillo-Franco M. and Mirbach B., "Vehicle occupancy monitoring with optical range-sensors", IEEE Intelligent Vehicles Symposium 2004, June 2004, pp90 – 94.

[03] Devarakota, P.R.R.; Mirbach, B.; Castillo-Franco, M.; Ottersten, "3D vision technology for occupant detection and classification; B". Fifth International Conference on 3-D Digital Imaging and Modeling, 2005. 3DIM 2005., 13-16 June 2005, pp72 – 79

[04] Dhond, U.R.; Aggarwal, J.K. "Structure from stereo-a review"., IEEE Transactions on Systems, Man and Cybernetics., Volume 19, Issue 6, Nov.-Dec. 1989, pp1489 – 1510

[05] Scharstein, D.; Szeliski, R., "High-accuracy stereo depth maps using structured light"., Computer Society Conference on Computer Vision and Pattern Recognition, 2003., Volume 1, 18-20 June 2003, ppI-195 - I-202 vol.1

[06] http://cat.middlebury.edu/stereo

[07] Nedevschi, S.; Danescu, R.; Frentiu, D.; Marita, T.; Oniga, F.; Pocol, C.; Schmidt, R.; Graf, T., "High accuracy stereo vision system for far distance obstacle detection"., IEEE Intelligent Vehicles Symposium 2004, 14-17 June 2004, pp292 – 297

[08] Nedevschi, S.; Danescu, R.; Marita, T.; Oniga, F.; Pocol, C.; Sobol, S., Graf, T.; Schmidt, R., "Driving environment perception using stereovision".,; IEEE Intelligent Vehicles Symposium, 2005. Proceedings., 6-8 June 2005, pp331 – 336

[09] Q Yu, H Araújo, H Wang., "A Stereovision Method for Obstacle Detection and Tracking in Non-Flat Urban Environments"., Springer Valley Autonomous Robots, 2005

[10] Nedevschi, S.; Schmidt, R.; Graf, T.; Danescu, R.; Frentiu, D.; Marita, T.; Oniga, F.; Pocol, C., "3D lane detection system based on stereovision"., IEEE Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference, 3-6 Oct. 2004, pp161 – 166

[11] Matuszyk, L., Zelinsky, A., Nilsson, L., Rilbe, M., "Stereo panoramic vision for monitoring vehicle blind-spots", IEEE Intelligent Vehicles Symposium 14-17, 2004, June 2004, pp31 – 36

[12] Suzuki, Y., Koyamaishi, M., Yendo, T.; Fujii, T., Tanimoto, M.Parking, "assistance using multi-camera infrastructure", IEEE Intelligent Vehicles Symposium, 2005. Proceedings., 6-8 June 2005, pp106 – 111

[13] J Salvi, X Armangue, J Batlle, "A comparative review of camera calibrating methods with accuracy evaluation", - Pattern Recognition, 2002, Volume 35, pp 1617-1635

[14] SJ Maybank, OD Faugeras, "A theory of self-calibration of a moving camera", International Journal of Computer Vision, 1992, Volume 8 , Issue 2, pp123 - 151

[15] Model Selection, C Matsunaga, K Kanatani, "Calibration of a Moving Camera Using a Planar Pattern: Optimal Computation, Reliability Evaluation, and Stabilization", ECCV (2), 2000, Volume 1843 / 2000, pp595 - 609

[16] Starck, J., Hilton, A., "Scene reconstruction from multiple cameras", Ninth IEEE International Conference on Computer Vision, 2003 Proceedings., 13-16 Oct. 2003, pp915 – 922, vol.2

[17] Tao Zhao, Aggarwal, M., Kumar, R., Sawhney, H., "Real-time wide area multi-camera stereo tracking", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005., Volume 1, 20-25 June 2005, pp976 – 983

[18] Saito, H., Baba, S., Kimura, M., Vedula, S., Kanade, T., "Appearance-based virtual view generation of temporally-varying events from multi-camera images in the 3D room", IEEE Second International Conference on 3-D Digital Imaging and Modeling, 1999. Proceedings., 4-8 Oct. 1999, pp516 – 525

[19] Heitz, F.; Bouthemy, P., "Multimodal motion estimation and segmentation using Markov random fields", IEEE 10th International Conference on Pattern Recognition, 1990. Proceedings., Volume 1, 16-21 June 1990, pp378 - 383 vol.1

[20] Jie Wei, Ze-Nian Li, "An efficient two-pass MAP-MRF algorithm for motion estimation based on mean field theory", IEEE Transactions on Circuits and Systems for Video Technology, Volume 9, Issue 6, Sept. 1999, pp960 – 972

[21] Bao Hongqiang, Zhang Zhaoyang, "Object-based video segmentation using spatio-temporal energy", ICSP '04. 2004 7th International Conference on Signal Processing, 2004. Proceedings., Volume 2, 31 Aug.-4 Sept. 2004, pp:1260 - 1263

[22] Wei Zeng, Wen Gao, "Accurate moving object segmentation by a hierarchical region labeling approach", IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)., Volume 3, 17-21 May 2004 ppiii - 637-40

[23] R Labayrade, D Aubert, JP Tarel, "Real Time Obstacle Detection in Stereo Vision on non Flat Road Geometry through 'V-Disparity'", - Procs. IEEE Intelligent Vehicles Symposium, 2002

[24] Hu, Z., Uchimura, K., "U-V-disparity: an efficient algorithm for stereovision based scene analysis", IEEE Intelligent Vehicles Symposium, 2005. Proceedings., 6-8 June 2005 pp48 – 54

[25] Broggi, A., Caraffi, C., Fedriga, R.I., Grisleri, P., "Obstacle Detection with Stereo Vision for Off-Road Vehicle Navigation", 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Volume 3, 20-26 June 2005 pp65 – 65

[26] Rebut, J., Toulminet, G., Bensrhair, A., "Road obstacles detection using a self-adaptive stereo vision sensor: a contribution to the ARCOS French project", 2004 IEEE Intelligent Vehicles Symposium, 14-17 June 2004 pp738 – 743

[27] R Labayrade, D Aubert, "Robust and Fast Stereovision Based Road Obstacles Detection for Driving Safety Assistance", - IAPPR Workshop on Machine Vision, Applications, Nara, Japan, 2002

[28] R Labayrade, D Aubert, "In-vehicle obstacles detection and characterization by stereovision", - IEEE In Vehicle Cognitive Computer Vision Systems, 2003

[29] Michael, Isard, Andrew, Blake, "CONDENSATION-Conditional Density Propagation for Visual Tracking", International Journal of Computer Vision, Volume 29, Number 1, August 1998, pp5 – 28

[30] Esther, B. Koller-Meier, Frank, Ade, "Tracking Multiple objects Using the Condensation Algorithm", Robotics and Autonomous Systems, Volume 34, 2001, pp 93 – 105

[31] Simon Maskell, Malcolm Rollason, Neil Gordon, David Salmond, "Efficient Particle Filtering for Multiple Target Tracking with Application to Tracking in Structured Images", Image and Vision Volume 21, 2003, pp931 – 939

[32] Doucet, A., Vo, B.-N., Andrieu, C., Davy, M., "Particle Filtering for Multi-target Tracking and Sensor Management", IEEE Proceedings of the Fifth International Conference on Information Fusion, 2002., Volume 1, 8-11 July 2002, pp 474 - 481 vol.1

[33] Katja Nummiaro, Esther Koller-Meier, Luc Van Gool, "Object Tracking with an Adaptive Color-Based Particle Filter", Proceedings of the 24th DAGM Symposium on Pattern Recognition, 2002, pp353 – 360

[34] F. Aherne, N. Thacker and P. Rockett, Kybernetika, "The Bhattacharyya Metric as an Absolute Similarity Measure for Frequency Coded Data", vol. 34, no. 4, pp 363-368, 1998

[35] Kailath, T., "The Divergence and Bhattacharyya Distance Measures in Signal Selection", IEEE Transactions on Communications, [legacy, pre – 1988], Volume 15, Issue 1, Feb 1967, pp52 – 60

[36] Xia Limin, "Object tracking using color-based Kalman particle filters", ICSP '04. 2004 7th International Conference on Signal Processing, 2004 Proceedings., Volume 1, 31 Aug.-4 Sept. 2004 pp679 - 682 vol.1

[37] Perez, P., Vermaak, J., Blake, A., "Data Fusion for Visual Tracking with Particles", Proceedings of the IEEE, Volume 92, Issue 3, Mar 2004, pp 495 – 513

[38] Kuchi, P., Panchanathan, S., "On the use of joint estimation in particle filters for object tracking in video", TENCON 2004. 2004 IEEE Region 10 Conference, Volume A, 21-24 Nov. 2004 pp463 - 466 Vol. 1

[39] Siyuan Xing, Hongbing Ji, "A track-before-detect algorithm based on particle filter with model estimation", ICSP '04. 2004 7th International Conference on Signal Processing, 2004 Proceedings., Volume 1, 31 Aug.-4 Sept. pp311 - 314 vol.1

[40] Edengren, K.H., "Decomposition of edge operators", IEEE 9th International Conference on Pattern Recognition, , 14-17 Nov. 1988 pp963 - 965 vol.2

[41] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/

[42] Wong, S., Vassiliadis, S., Cotofana, S.A, "sum of absolute differences implementation in FPGA hardware", Euromicro Conference, 2002. Proceedings. 28th

[43] Hariyama, M., Yokoyama, N., Kameyama, M., Kobayashi, Y., "FPGA implementation of a stereo matching processor based on window-parallel-and-pixel-parallel architecture", 48th Midwest Symposium on Circuits and Systems, 2005.

[44] Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T., "A Tutorial on Particle Filters for Online Non-linear/Non-Gaussian Bayesian Tracking", IEEE Transaction on Signal Processing, Volume 50, Issue 2, Feb. 2002, pp174 - 188

[45] Perez, P., Vermaak, J., Blake, A., "Data Fusion for Visual Tracking with Particles", Proceedings of the IEEE, Volume 92, Issue 3, Mar 2004, pp 495 – 513

[46] Sun, Z., Bebis, G., Miller, R., "On-road vehicle detection using optical sensors: a review", The 7th International IEEE Conference on Intelligent Transportation Systems, 2004 Proceedings., 3-6 Oct. 2004 pp585 - 590

[47]http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ISARD1/condensation.html

[48] http://dit.unitn.it/~diplodoc/

[49] Hartley R. I., "Kruppa's Equations Derived from the Fundamental Matrix", IEEE Transaction on Pattern Analysis and Machine Intelligence, February 1997 (Vol. 19, No. 2), pp133-135

[50] Zhou Chuan, Tan Da Long, Zhu Feng, Dong Zai Li, "A planar homography estimation method for camera calibration", 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003 Proceedings., Volume 1, 16-20 July 2003 pp424 - 429 vol.1

[51] http://users.rcn.com/mclaughl.dnai/

[52] Papoulis, A., "Probability and Statistics". Pretice-Hall., 1990

[53] Grenander, U., Chow, Y., and Keenan, D. M., "A Pattern Theoretical Study of Biological Shapes"., Springer-Verlag., New York., 1991

[54] Gelb, A., ed., "Applied Optimal Estimation"., MIT Press, Cambridge, MA., 1984

[55] Stan Z. Li, "Markov Random Field, Modeling in Computer Vision", Springer-Verlag 1995

[56] Takagi, K., Morikawa, K., Ogawa, T., Saburi, M., "Road Environment Recognition Using On-vehicle LIDAR", 2006 IEEE Intelligent Vehicles Symposium, 13-15 June 2006, pp120- 125

[57] Guang Lu, Masayoshi Tomizuka, "LIDAR Sensing for Vehicle Lateral Guidance: Algorithm and Experimental Study", IEEE/ASME Transactions on Mechatronics, Volume: 11, Issue: 6, Dec. 2006, pp653-660

[58] Mahlisch, M., Schweiger, R., Ritter, W., Dietmayer, K., "Sensorfusion Using Spatio-Temporal Aligned Video and Lidar for Improved Vehicle Detection", 2006 IEEE Intelligent Vehicles Symposium, 13-15 June 2006, pp424- 429

[59] ECOVISION project main page,

http://www.pspc.dibe.unige.it/ecovision/index.html

[60] ECOVISION Periodic Progress Report N°:3,

http://www.pspc.dibe.unige.it/ecovision/pubs/

[61] Krüger, N., & Felsberg, M. (2004). An explicit and compact coding of geometric and structural information applied to stereomatching. Pattern Recognition Letters, 25(8):849-863, 2004.

[62] M. Felsberg and G. Sommer. The monogenic signal. IEEE Transactions on Signal Processing, 41(12), 2001

[63] A. Sikora, Riesz transform, Gaussian bounds and the method of wave equation. Math. Z. 247 (2004)

[64] Pugeault, N. & Krüger, N. (2003). Multi--modal Matching applied to Stereo. Proceedings of the British Machine Vision Conference, 2003

[65] Pugeault, N., Wörgötter, F. & Krüger, N. (2004). A non-local Stereo Similarity based on Collinear Groups. Proceedings of the Fourth International ICSC Symposium on ENGINEERING OF INTELLIGENT SYSTEMS, 2004

[66] Kovesi P. D., MATLAB and Octave Functions for Computer Vision and Image Processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from:

http://www.csse.uwa.edu.au/~pk/research/matlabfns/

[67] Kovesi. P. D., MATLAB and Octave Functions for Computer Vision and Image Processing, http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/Match

/matchbymonogenicphase.m

[68] C Harris, M Stephens, A combined corner and edge detector, Alvey Vision Conference, 1988

[69] Wong, S.; Vassiliadis, S.; Cotofana, S.; A sum of absolute differences implementation in FPGA hardware; Euromicro Conference, 2002. Proceedings. 28th; 4-6 Sept. 2002 Page(s):183 – 188

[70] IceRobotics, Home Page: http://www.icerobotics.com/

[71] Camera Calibration Toolbox for Matlab:

http://www.vision.caltech.edu/bouguetj/calib_doc/

[72] Kato Z., Supervised Image Segmentation Using Markov Random Fields, http://www.inf.u-szeged.hu/~kato/software/mrfdemo.html

[73] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. and Teller E., Equation of state calculations by fast computing machines, J. Chem. Physics, 21 (1953) 1087-1092

[74] Geman S. and Geman D., Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, IEEE Trans. Patt. Analysis and Mach. Intel., 6 (1984) 721-741

[75] Jeng F.C.and Woods J.M., Compound Gauss-Markov Random Fields for image estimation, IEEE Trans. Acoust., Speech and Signal Proc., 39 (1991) 638-687

[76] Besag J., On the statistical analysis of dirty pictures, J. Roy. Statis. Soc. B., 68 (1986) 259-302

[77] Berthod M, Kato Z, Yu S, and Zerubia J. Bayesian Image Classification Using Markov Random Fields. Image and Vision Computing, 14:285--295, 1996

[78] Middlebury stereo vision page: http://vision.middlebury.edu/stereo/

[79] Bob Fisher, Computer Vision, Department of Computer Science, The University of Edinburgh,

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT10/node3.html

Appendix

Programming and code

As the algorithms are described in chapter 4, the code of the final approach is described in detail here. The coding and programming were developed in Matlab. The whole tracking code consists of 18 Matlab executable M-files. A list of these is as follow:

CD_Diplo
CD_init_Model
CD_VehicleDetection_init
CD_init_Default
CD_run
CD_VehicleDetection
CD_HorizontalMotion
CD_VehicleStatus
CD_ParticleGeneration
CD_Obtain_observation
CD_Bhattacharry
CD_Calculate_base_weights
CD_Evaluate_observation_density
CD_Predict_sample_position
CD_Predict_new_base
CD_Pick_base_sample
CD_Update_after_iteration
CD_Display

1 Global Data

startF	Int, The first frame number of the processing sequence
endF	Int, The last frame number of the processing sequence
Niterations	Int, Number of time steps, simply startF-endF
Nsamples	Int, Number of particles used in the condensation algorithm
imdir	String, Directory of where the images are stored
disdir	String, Directory of where the disparity maps are stored
im	Matrix, Matrix for hold the right image in the current time step
dis	Matrix, Matrix for hold the disparity map in the current time step
process_sigma	2×1 vector, sigma of Gaussian distribution horizontally and vertically

density_sigma	Double, sigma of observation density evaluation
Ncar	Int, number of vehicles at each time step
object	$6 \times Ncar$ matrix, holding the dispairy, vertical position, horizontal position, width, height and image occlusion flag for detected vehicles at each time step
chist_previous	$Ncar \times 512$ matrix, each row holding the colour histogram distribution of the detected vehicle at each time step
data_measured_position	$2 \times Niterations \times Ncar$ matrix, holding the whole measured positions for all vehicles of all time steps
drange	$2 \times Ncar$ matrix, holding the disparity range for detected vehicles at each time step
data_old_positions	$2 \times Nsamples \times Ncar$ matrix, holding the particles before sampling at each time step
data_new_positions	$2 \times Nsamples \times Ncar$ matrix, holding the particles after sampling at each time step
data_sample_weights	$Nsamples \times Ncar$ matrix, holding the weights of the old particles before sampling at each time step
data_cumul_prob_array	$Nsamples \times Ncar$ matrix, holding the accumulated weights of particles at each time step
data_largest_cumul_prob	$Ncar \times 1$ vector, holding the largest accumulated weight at each time step
data_observation	$Nsamples \times Ncar$ matrix, holding the Bhattacharyya coefficients of particles at each time step
data_newsample_weights	$Nsamples \times Ncar$ matrix, holding the weights of the new particles after sampling at each time step

2 Function Description

```
function CD_Diplo(StartF,EndF)

global startF
global endF

endF=EndF;
startF=StartF;
CD_init_Model;
CD_init_Default;
CD_run;
```

This is the main function which starts the whole condensation tracking. It takes two inputs, startF and endF, which define the first and last frame of the sequence to be processed. It also calls CD_init_Model, CD_init_Default and CD_run.

```
function CD_init_Model

global Niterations
global startF
global endF
global imdir
global disdir
global im
global dis
global imbase
global imname
global Ncar
global object
global chist_previous
global data_measured_position
global Nsamples
global drange

Nsamples=100;
Niterations=endF-startF;
imdir='~/matlab/Images/DIPLO/';
disdir='~/matlab/Results/diplodoc/New/';
imbase='IMG';
drange=zeros(2,2);
data_measured_position=zeros(2,Niterations+3,2);
chist_previous=zeros(2,512);
for i=3:-1:1
im=imread(['~/matlab/Images/DIPLO/IMG0000' num2str(startF-i) '_R.PNG']);
dis=imread(['~/matlab/Results/diplodoc/New/IMG0000' num2str(startF-i) '_d.png']);
[Ncar d x y chist ra rb]=CD_VehicleDetection_init;
object=zeros(6,Ncar);
object(1,:)=d;
object(2,:)=x;
object(3,:)=y;
object(4,:)=ra;
object(5,:)=rb;
data_measured_position(1,i,:)=x;
data_measured_position(2,i,:)=y;
end
local_im_display(im,2,100,100,['Initial Frame']);
for i=1:Ncar
   line([data_measured_position(2,3,i) data_measured_position(2,3,i)], ...
      [data_measured_position(1,3,i)-object(5,i)*object(1,i)
data_measured_position...
(1,3,i)],'Color',[1,0,0])
line([data_measured_position(2,3,i)-object(4,i)*object(1,i) data_measured_position...
(2,3,i)-object(4,i)*object(1,i)],[data_measured_position(1,3,i)-...
object(5,i)*object(1,i) data_measured_position(1,3,i)],'Color',[1,0,0])
```

```
line([data_measured_position(2,3,i) data_measured_position(2,3,i)-...
object(4,i)*object(1,i)],   [data_measured_position(1,3,i) data_measured_position...
(1,3,i)],'Color',[1,0,0])
line([data_measured_position(2,3,i) data_measured_position(2,3,i)-...
object(4,i)*object(1,i)],[data_measured_position(1,3,i)-object(5,i)*object(1,i)...
data_measured_position(1,3,i)-object(5,i)*object(1,i)],'Color',[1,0,0])
end
```

The function CD_init_Model initializes all the data related to the particular vehicle model. It also detects the vehicles and finds the positions for them in the first three frames by calling CD_VehicleDetection_init.

```
function [tNcar td tx ty tchist tra trb]=CD_VehicleDetection_init

global dis
global im

vdis=vdisp(dis,5,25);
udis=udisp(dis,5,25);
vdis=mythresh(vdis,35);
udis=mythresh(udis,27);
vdis(:,1)=0;
udis(:,1)=0;
[vr vc]=size(vdis);
tNcar=0;
td=[];
tx=[];
ty=[];
tchist=[];
tdhist=[];
ta=[];
tb=[];
for i=vc:-1:2
   pv=find(vdis(:,i));
   npv=length(pv);
   if npv>=20;
      tNcar=tNcar+1;
      td=[td i+4];
      tx=[tx pv(npv)-6];
      tb=[tb pv(npv)-4-pv(1)];
   end
end
for i=td
   pu=find(udis(1:200,i-4));
   if length(ty)==0
```

```
      ty=[ty pu(length(pu))-3];
      ta=[ta pu(length(pu))-pu(1)-4];
   else
      pv=find(vdis(:,i-4));
      npv=length(pv);
      ta=[ta pv(npv)-pv(1)-4];
      ty=[ty pu(length(pu))-3];
   end
end
for i=1:tNcar
   region=im(tx(i):tx(i)+tb(i),ty(i)-ta(i):ty(i),:);
   chist=invhist(region,1);
   tchist(i,:)=chist(:)/sum(chist(:))';
end
tra=double(ta./td);
trb=double(tb./td);
```

The function CD_VehicleDetection_init detects vehicles in the current image. It finds the disparity, horizontal position, vertical position, width, height, and calculates the colour histogram for the detected vehicles. It returns all those results as output.

```
function CD_init_Default

global process_sigma
global density_sigma
global Nsamples
global Ncar
global data_old_positions
global data_new_positions
global data_sample_weights
global data_cumul_prob_array
global data_largest_cumul_prob
global data_observation
global data_newsample_weights

process_sigma=[10;10];
density_sigma=0.05;
data_old_positions=zeros(2,Nsamples,Ncar);
data_new_positions=zeros(2,Nsamples,Ncar);
data_sample_weights=zeros(Nsamples,Ncar);
data_cumul_prob_array=zeros(Nsamples,Ncar);
data_largest_cumul_prob=zeros(1,Ncar);
data_observation=zeros(2,Nsamples,Ncar);
data_newsample_weights=zeros(Nsamples,Ncar);
```

The function CD_init_Default initializes all the data and parameters needed for the condensation filter.

```
function CD_run

global imname
global im
global dis
global Niterations
global imbase
global startF
global imdir
global disdir
global ii
global Ncar
global object

for ii=1:Niterations
   imname=[imbase '0000' num2str(startF+ii)];
   im=imread([imdir imname '_R.PNG']);
   dis=double(imread([disdir imname '_d.png']));
   CD_VehicleDetection;
   CD_ParticleGeneration;
   CD_obtain_observations;
   CD_calculate_base_weights;
   CD_predict_new_bases;
   CD_update_after_iteration(ii);
End
```

The function CD_run runs the condensation filter. It calls CD_VehicleDetection, CD_ParticleGeneration, CD_obtain_observations, CD_calculate_base_weights, CD_predict_new_bases, and CD_update_after_iteration.

```
function CD_VehicleDetection

global object
global data_measured_position
global ii
global dis
global Ncar
global drange
```

```
vdis=vdisp(dis,5,25);
udis=udisp(dis,5,25);
vdis=mythresh(vdis,30);
udis=mythresh(udis,27);
vdis(:,1)=0;
udis(:,1)=0;
[ur uc]=size(udis);
vd=zeros(1,Ncar);
ud=zeros(1,Ncar);
ulower=zeros(1,Ncar);
vupper=zeros(1,Ncar);
u_p=zeros(1,Ncar);
tt=[26 1];
t=zeros(1,Ncar);
dd=zeros(1,Ncar);
ncar=Ncar;
for i=1:Ncar
  if object(6,i)==0
    u_p(i)=CD_UM(i);
    u1=round(u_p(i)-object(4,i)*object(1,i))-10;
    if u1<1
      u1=1;
    end
    u2=u_p(i)+5;
    if u2>300
      u2=300;
    end
    npu=zeros(1,drange(2,i)-drange(1,i)+3);
    for j=1:drange(2,i)-drange(1,i)+3
      npu(j)=length(find(udis(u1:u2,drange(1,i)-6+j)));
    end
    tnpu=(npu>=6);
    npu=npu.*tnpu;
    [uC uI]=max(npu);
    for j=uI:length(npu)
      if npu(j)==0
        npu(j:length(npu))=0;
        break
      end
    end
    for j=uI:-1:1
      if npu(j)==0
        npu(1:j)=0;
        break
      end
    end
    dsum=0;
    for j=1:length(npu)
```

```
        dsum=dsum+(j+drange(1,i)-2)*npu(j);
    end
    dd(i)=double(dsum/sum(npu));
    object(1,i)=dd(i);
    dpu=find(npu);
    drange(1,i)=min(dpu)+drange(1,i)-2;
    drange(2,i)=length(dpu)+drange(1,i)-1;
elseif object(6,i)==1
    u1=round(data_measured_position(2,ii+2,i));
    u2=round(data_measured_position(2,ii+2,i)+object(4,i)*object(1,i))+10;
    if u2>320
        u2=320;
    end
    npu=zeros(1,drange(2,i)-drange(1,i)+3);
    for j=1:drange(2,i)-drange(1,i)+3
        npu(j)=length(find(udis(u1:u2,drange(1,i)-6+j)));
    end
    tnpu=(npu>=6);
    npu=npu.*tnpu;
    [uC uI]=max(npu);
    for j=uI:length(npu)
        if npu(j)==0
            npu(j:length(npu))=0;
            break
        end
    end
    for j=uI:-1:1
        if npu(j)==0
            npu(1:j)=0;
            break
        end
    end
    dsum=0;
    for j=1:length(npu)
        dsum=dsum+(j+drange(1,i)-2)*npu(j);
    end
    dd(i)=double(dsum/sum(npu));
    object(1,i)=dd(i);
    dpu=find(npu);
    drange(1,i)=min(dpu)+drange(1,i)-2;
    drange(2,i)=length(dpu)+drange(1,i)-1;
else
    object(2,i)=240;
    object(3,i)=320;
    dd(i)=0;
    object(1,i)=dd(i);
    object(6,i)=2;
end
```

```
end
CD_VehicleStatus(u_p,dd,udis,vdis);
```

As explained in Chapter 4, the function CD_VehicleDetection detects the vehicles and finds the disparity and disparity range for them. It calls CD_HorizontalMotion to find a better non-zero points counting area. It calls CD_VehicleStatus to find the status and calculate the positions for the detected vehicles.

```
function [u_pp]=CD_HorizontalMotion(it)

global data_measured_position
global ii
global object

a=2*data_measured_position(2,ii+1,it)-data_measured_position(2,ii+2,it)...
  -data_measured_position(2,ii,it);
if a>10
  u_pp=2*data_measured_position(2,ii+2,it)-data_measured_position(2,ii+1,it)...
    +a;
else
  u_pp=data_measured_position(2,ii+2,it)+5;
end
```

This function finds the horizontal position for the vehicle according to the horizontal motion model and returns the estimated horizontal position as output.

```
function CD_VehicleStatus(su_p,sdd,sudis,svdis);

global object
global data_measured_position
global ii
global dis
global Ncar
global drange

tt=[10 1 10];
as=[2 2 5];
st=zeros(1,Ncar);
for i=1:Ncar
  if object(6,i)==0
    if sdd(i)~=floor(sdd(i))
      xp1=find(svdis(round(data_measured_position(1,ii+2,i))-10:round...
```

```
        (data_measured_position(1,ii+2,i))+10,floor(sdd(i))-4));
      xp2=find(svdis(round(data_measured_position(1,ii+2,i))-10:round...
        (data_measured_position(1,ii+2,i))+10,ceil(sdd(i))-4));
      x1=xp1(length(xp1))+round(data_measured_position(1,ii+2,i))-10;
      x2=xp2(length(xp2))+round(data_measured_position(1,ii+2,i))-10;
      object(2,i)=x1*(sdd(i)-floor(sdd(i)))+x2*(ceil(sdd(i))-sdd(i))-8;
    else
      xp=find(svdis(round(data_measured_position(1,ii+2,i))-10:round...
        (data_measured_position(1,ii+2,i))+10,sdd(i)-4));
      x=xp(length(xp))+round(data_measured_position(1,ii+2,i))-10;
      object(2,i)=x-8;
    end
    u1=round(su_p(i)-object(4,i)*object(1,i))-10;
    if u1<1
      u1=1;
    end
    u2=su_p(i)+5;
    if u2>300
      u2=300;
    end
    cudis=sudis;
    npa=length(find(cudis(u1:u2,drange(1,i)-4:drange(2,i)-4)));
    npc=length(find(cudis(u1:u2,round(sdd(i)-4))));
    npt=npa-npc;
    if npt>=tt(i)
      st(i)=1;
      if object(4,i)<13
        object(4,i)=object(4,i)+double((su_p(i)-data_measured_position...
          (2,ii+2,i))/object(1,i)/as(i));
      end
    else
      st(i)=0;
    end
    [ux uy]=find(cudis(u1:u2,drange(1,i)-5:drange(2,i)-4));
    if u2~=300
      ulower=max(ux)+u1;
      object(3,i)=ulower;
    else
      uupper=min(ux)+u1;
      object(3,i)=uupper;
      object(6,i)=1;
    end
  elseif object(6,i)==1
    u1=round(data_measured_position(2,ii+2,i));
    u2=round(data_measured_position(2,ii+2,i)+object(4,i)*object(1,i))+10;
    if u2>320
      u2=320;
    end
```

```
      cudis=sudis;
      npa=length(find(cudis(u1:u2,drange(1,i)-5:drange(2,i)-4)));
      npc=length(find(cudis(u1:u2,round(sdd(i)-4))));
      npt=npa-npc;
      if npt>=tt(i)
         st(i)=1;
      else
         st(i)=0;
      end
      [ux uy]=find(cudis(u1:u2,drange(1,i)-4:drange(2,i)-4));
      uupper=min(ux)+u1;
      if uupper<300
         object(3,i)=uupper;
         object(6,i)=1;
      else
         object(3,i)=320;
         object(6,i)=2;
      end
   else
      object(3,i)=320;
      object(6,i)=2;
   end
end
```

The function CD_VehicleStatus decides the status of the detected vehicles, finds the positions for them and amends the width for them. It also decides if the vehicle is occluded by the image. The totally occluded (object(6,i)=2), partially occluded (object(6,i)=1) and non-occluded (object(6,i)=0) vehicles are treated differently in the program.

```
function CD_ParticleGeneration

global data_old_positions
global Nsamples
global process_sigma
global object
global Ncar
global ii

for j=1:Ncar
   random1=randn(1,Nsamples);
   random1=random1/max(random1);
   random2=randn(1,Nsamples/2);
random2=random2/max(random2);
```

```
random3=randn(1,Nsamples/2);
random3=random3/max(random3);
   data_old_positions(1,:,j)=object(2,j)+process_sigma(1,1)*random1;
   data_old_positions(2,1: Nsamples/2,j)=object(3,j)+process_sigma(2,1)*random2;
   x_p=CD_HorizontalMotion(j);
   data_old_positions(2,
Nsamples/2+1:Nsamples,j)=x_p+process_sigma(2,1)*random3;
end
```

This function generates the particles for the detected vehicle positions. As mentioned in Chapter 4, the vertical particles are generated by Gaussian distribution using the detected position and horizontal ones by a combination of detected position and motion model estimation (half and half in the program).

```
function CD_obtain_observations

CD_Bhattacharyya;
```

It calls function CD_Bhattacharyya.

```
function CD_Bhattacharyya

global Nsamples
global data_old_positions
global chist_previous
global data_observation
global imseg
global disseg
global Ncar
global object

for i=1:Ncar
   for n=1:Nsamples
      if object(6,i)==0
         u1=round(data_old_positions(2,n,i)-object(4,i)*object(1,i));
         if u1<1
            u1=1;
         end
         region=im(round(data_old_positions(1,n,i)-object(5,i)*object(1,i)):...
round(data_old_positions(1,n,i)),u1:round(data_old_positions(2,n,i)),:);
         hist=invhist(region,1);
         histogram=hist(:)/sum(hist(:));
         sq=sqrt(histogram'.*chist_previous(i,:));
```

```
        rou=sum(sq);
        BD=sqrt(1-rou);
        data_observation(1,n,i)=BD;
      elseif object(6,i)==1
        u2=round(data_old_positions(2,n,i)+object(4,i)*object(1,i));
        if u2>320
          u2=320;
        end
        region=imseg(round(data_old_positions(1,n,i)-object(5,i)*object(1,i)):...
round(data_old_positions(1,n,i)),round(data_old_positions(2,n,i)):u2,:);
        hist=invhist(region,1);
        histogram=hist(:)/sum(hist(:));
        sq=sqrt(histogram'.*chist_previous(i,:));
        rou=sum(sq);
        BD=sqrt(1-rou);
        data_observation(1,n,i)=BD;
      end
    end
end
```

The function CD_ Bhattacharyya calculates the Bhattacharyya coefficient for each particle.

```
function CD_calculate_base_weights

global Nsamples
global data_sample_weights
global data_cumul_prob_array
global data_largest_cumul_prob
global data_observation
global Ncar
global object

for i=1:Ncar
  if object(6,i)~=2
    cumul_total=0;
    for n=1:Nsamples
      data_sample_weights(n,i)=CD_evaluate_observation_density...
        (data_observation(1,n,i));
      data_cumul_prob_array(n,i)=cumul_total;
      cumul_total=cumul_total+data_sample_weights(n,i);
    end
    data_largest_cumul_prob(i)=cumul_total;

data_cumul_prob_array(:,i)=data_cumul_prob_array(:,i)/data_largest_cumul_prob(i);
```

```
data_sample_weights(:,i)=data_sample_weights(:,i)/data_largest_cumul_prob(i);
      data_largest_cumul_prob(i)=1;
   end
end
```

The weights of the particles are calculated by calling CD_evaluate_observation_density in this function. Then the weights are normalized.

```
function p=CD_evaluate_observation_density(observation1)

global density_sigma

p=1/(sqrt(2*pi)*density_sigma)*exp...
   (-0.5*observation1^2/density_sigma^2);
```

This function evaluates the probability according to $p = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{(1-\rho)}{2\sigma^2}}$.

```
function CD_predict_new_bases

global Nsamples
global Ncar
global object

for i=1:Ncar
   if object(6,i)~=2
      for n=1:Nsamples
         base=CD_pick_base_sample(i);
         CD_predict_sample_position(n,base,i);
      end
   end
end
```

This function samples the particles and generates a new set of particles. The old particles are picked by calling CD_pick_base_sample and the new particles are generated by calling CD_predict_sample_position.

```
function base=CD_pick_base_sample(i)

global data_largest_cumul_prob
```

```
global Nsamples
global data_cumul_prob_array

choice=rand(1)*data_largest_cumul_prob(i);
low=1;
high=Nsamples;
while high>low+1
   middle=uint8(high+low)/2;
   if choice>data_cumul_prob_array(middle,i)
      low=middle;
   else
      high=middle;
   end
end
base=low;
```

Particles are picked by a stochastic binary search in this function.

```
function CD_predict_sample_position(new_sample,old_sample,i)

global data_new_positions
global data_old_positions
global data_newsample_weights
global data_sample_weights

data_new_positions(:,new_sample,i)=data_old_positions(:,old_sample,i);
data_newsample_weights(new_sample,i)=data_sample_weights(new_sample,i);
```

The picked particles are filled in the data_new_positions and their weights in data_newsample_weights.

```
function CD_update_after_iteration(iter)

global data_new_positions
global data_old_positions
global data_newsample_weights
global data_sample_weights

CD_display(iter);
```

Final positions of the detected vehicles are calculated in CD_display. Bounding boxes are also drawn on the image in CD_display. Colour histograms of the bounding boxes are calculated for comparison at the next time step.

```
function CD_display(iter)

global data_old_positions
global data_new_positions
global im
global dis
global data_measured_position
global Nsamples
global data_newsample_weights
global data_largest_cumul_prob
global object
global Ncar
global chist_previous
global startF
global drange

aggregate=zeros(2,Ncar);
for i=1:Ncar

data_newsample_weights(:,i)=data_newsample_weights(:,i)/sum(data_newsample_weights(:,i));
   aggregate(1,i)=sum(data_new_positions(1,:,i).*data_newsample_weights(:,i)');
   aggregate(2,i)=sum(data_new_positions(2,:,i).*data_newsample_weights(:,i)');
data_measured_position(:,iter+3,i)=aggregate(:,i);
   if object(6,i)==0
      u1=round(data_measured_position(2,iter+3,i)-object(4,i)*object(1,i));
      if u1<1
         u1=1;
      end
      clear region
      region=im(round(data_measured_position(1,iter+3,i),object(5,i)*object(1,i)):...
round(data_measured_position(1,iter+3,i)),u1:round(data_measured_position...
(2,iter+3,i)),:);
      hist=invhist(region,1);
      chist_previous(i,:)=hist(:)/sum(hist(:));
   elseif object(6,i)==1
      clear region
      u2=round(data_measured_position(2,iter+3,i)+object(4,i)*object(1,i));
      if u2>320
         u2=320;
      end
      region=im(round(data_measured_position(1,iter+3,i)-object(5,i)*object(1,i)):...
round(data_measured_position(1,iter+3,i)),round(data_measured_position...
(2,iter+3,i)):u2,:);
      hist=invhist(region,1);
      chist_previous(i,:)=hist(:)/sum(hist(:));
   end
```

```
end
xpos=100;
ypos=100;
local_im_display(im,2,xpos,ypos,['Frame' num2str(startF+iter)]);
for i=1:Ncar
  if object(6,i)==0
    u1=data_measured_position(2,iter+3,i)-object(4,i)*object(1,i);
    if u1<1
      u1=1;
    end
    line([data_measured_position(2,iter+3,i) data_measured_position(2,iter+3,i)], ...
      [data_measured_position(1,iter+3,i)-object(5,i)*object(1,i) data_measured...
_position(1,iter+3,i)],'Color',circshift([1,0,0],[0 i-1]))
    line([u1 u1],[data_measured_position(1,iter+3,i)-object(5,i)*object(1,i)...
      data_measured_position(1,iter+3,i)],'Color',circshift([1,0,0],[0 i-1]))
    line([data_measured_position(2,iter+3,i) u1],[data_measured_position(1,iter+3,i)...
      data_measured_position(1,iter+3,i)],'Color',circshift([1,0,0],[0 i-1]))
    line([data_measured_position(2,iter+3,i) u1],[data_measured_position(1,iter+3,i)-...
object(5,i)*object(1,i) data_measured_position(1,iter+3,i)-object(5,i)...
*object(1,i)],'Color',circshift([1,0,0],[0 i-1]))
  elseif object(6,i)==1
    u2=data_measured_position(2,iter+3,i)+object(4,i)*object(1,i);
    if u2>320
      u2=320;
    end
    line([data_measured_position(2,iter+3,i) data_measured_position(2,iter+3,i)], ...
      [data_measured_position(1,iter+3,i)-object(5,i)*object(1,i) data_measured_...
position(1,iter+3,i)],'Color',circshift([1,0,0],[0 i-1]))
    line([u2 u2], [data_measured_position(1,iter+3,i)-object(5,i)*...
object(1,i) data_measured_position(1,iter+3,i)],...
'Color',circshift([1,0,0],[0 i-1]))
    line([data_measured_position(2,iter+3,i) u2],[data_measured_position(1,iter+3,i)...
      data_measured_position(1,iter+3,i)],'Color',circshift([1,0,0],[0 i-1]))
    line([data_measured_position(2,iter+3,i) u2],[data_measured_position(1,iter+3,i)-...
object(5,i)*object(1,i) data_measured_position(1,iter+3,i)...
-object(5,i)*object(1,i)],'Color',circshift([1,0,0],[0 i-1]))
  end
end
```